TECHNISCHE UNIVERSITÄT BERLIN
FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIK
LEHRSTUHL FÜR INTELLIGENTE NETZE UND
MANAGEMENT VERTEILTER SYSTEME

Impact of Location on Content Delivery

vorgelegt von
Bernhard Ager (Dipl.-Inf.)

von der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades
DOKTOR DER NATURWISSENSCHAFTEN (DR. RER. NAT.)
genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Jean-Pierre Seifert, TU Berlin
Gutachterin:	Prof. Anja Feldmann, Ph. D., TU Berlin
Gutachter:	Prof. Bruce Maggs, Ph. D., Duke University
Gutachter:	Prof. Dr. habil. Odej Kao, TU Berlin

Tag der wissenschaftlichen Aussprache: 25. Juni 2011

Berlin 2011
D83

Abstract

The increasing number of users as well as their demand for more and richer content has led to an exponential growth of Internet traffic for more than 15 years. In addition, new applications and use cases have changed the type of traffic. For example, social networking enables users to publish their own content. This *user generated content* is often published on popular sites such as YouTube, Twitter, and Facebook. Another example are the offerings of interactive and multi-media content by content providers, e. g., Google Maps or IPTV services. With the introduction of peer-to-peer (P2P) protocols in 1998 an even more radical change emerged because P2P protocols allow users to directly exchange large amounts of content: The peers transfer data without the need for an intermediary and often centralized server. However, as shown by recent studies Internet traffic is again dominated by HTTP, mostly at the expense of P2P.

This traffic growth increases the demands on the infrastructure components that form the Internet, e. g., servers and routers. Moreover, most of the traffic is generated by a few very popular services. The enormous demand for such popular content cannot be satisfied by the traditional hosting model in which content is located on a single server. Instead, content providers need to scale up their delivery infrastructure, e. g., by using replication in large data centers or by buying service from content delivery infrastructures, e. g., Akamai or Limelight. Moreover, not only content providers have to cope with the demand: The network infrastructure also needs to be constantly upgraded to keep up with the growing demand for content.

In this thesis we characterize the impact of content delivery on the network. We utilize data sets from both active and passive measurements. This allows us to cover a wide range of abstraction levels from a detailed protocol level view of several content delivery mechanisms to the high-level picture of identifying and mapping the content infrastructures that are hosting the most popular content.

We find that caching content is still hard and that the user's choice of DNS resolvers has a profound impact on the server selection mechanism of content distribution infrastructures. We propose Web content cartography to infer how content distribution infrastructures are deployed and what the role of different organizations in the Internet is. We conclude by putting our findings in the context of contemporary work and give recommendations on how to improve content delivery to all parties involved: users, Internet service providers, and content distribution infrastructures.

Zusammenfassung

Steigende Benutzerzahlen und steigende Internetnutzung sind seit über 15 Jahren verantwortlich für ein exponentielles Wachstum des Internetverkehrs. Darüber hinaus haben neue Applikationen und Anwendungsfälle zu einer Veränderung der Eigenschaften des Verkehrs geführt. Zum Beispiel erlauben soziale Netze dem Benutzer die Veröffentlichung eigener Inhalte. Diese *benutzergenerierten Inhalte* werden häufig auf beliebten Webseiten wie YouTube, Twitter oder Facebook publiziert. Weitere Beispiele sind die Angebote an interaktiven oder multimedialen Inhalten wie Google Maps oder Fernsehdienste (IPTV). Die Einführung von Peer-to-Peer-Protokollen (P2P) im Jahre 1998 bewirkte einen noch radikaleren Wandel, da sie den direkten Austausch von großen Mengen an Daten erlauben: Die Peers übertragen die Daten ohne einen dazwischenliegenden, oft zentralisierten Server. Allerdings zeigen aktuelle Forschungsarbeiten, dass Internetverkehr wieder von HTTP dominiert wird, zum Großteil auf Kosten von P2P.

Dieses Verkehrswachstum erhöht die Anforderungen an die Komponenten aus denen das Internet aufgebaut ist, z. B. Server und Router. Darüber hinaus wird der Großteil des Verkehrs von wenigen, sehr beliebten Diensten erzeugt. Die gewaltige Nachfrage nach solchen beliebten Inhalten kann nicht mehr durch das traditionelle Hostingmodell gedeckt werden, bei dem jeder Inhalt nur auf einem Server verfügbar gemacht wird. Stattdessen müssen Inhalteanbieter ihre Infrastruktur ausweiten, z. B. indem sie sie in großen Datenzentren vervielfältigen, oder indem sie den Dienst einer Content Distribution Infrastructure wie Akamai oder Limelight in Anspruch nehmen. Darüber hinaus müssen nicht nur die Anbieter von Inhalten sich der Nachfrage anpassen: Auch die Netzwerkinfrastruktur muss kontinuierlich mit der ständig steigenden Nachfrage mitwachsen.

In dieser Doktorarbeit charakterisieren wir die Auswirkung von Content Delivery auf das Netzwerk. Wir nutzen Datensätze aus aktiven und aus passiven Messungen, die es uns ermöglichen, das Problem auf verschiedenen Abstraktionsebenen zu untersuchen: vom detaillierten Verhalten auf der Protokollebene von verschiedenen Content Delivery-Methoden bis hin zum ganzheitlichen Bild des Identifizierens und Kartographierens der Content Distribution Infrastructures, die für die populärsten Inhalte verantwortlich sind.

Unsere Ergebnisse zeigen, dass das Cachen von Inhalten immer noch ein schwieriges Problem darstellt und dass die Wahl des DNS-Resolvers durch den Nutzer einen ausgeprägten Einfluß auf den Serverwahlmechanismus der Content Distribution Infrastructure hat. Wir schlagen vor, Webinhalte zu kartographieren, um darauf rückschließen zu können, wie Content Distribution Infrastructures

ausgerollt sind und welche Rollen verschiedene Organisationen im Internet einnehmen. Wir schließen die Arbeit ab, indem wir unsere Ergebnisse mit zeitnahen Arbeiten vergleichen und geben Empfehlungen, wie man die Auslieferung von Inhalten weiter verbessern kann, an alle betroffenen Parteien: Benutzer, Internetdienstanbieter und Content Distribution Infrastructures.

Contents

Papers

Published Papers

Parts of this thesis are based on the following peer-reviewed papers that have already been published.

Conferences

AGER, B., MÜHLBAUER, W., SMARAGDAKIS, G., AND UHLIG, S. Comparing DNS Resolvers in the Wild. In *Proceedings of the ACM Internet Measurement Conference* (2010).

AGER, B., SCHNEIDER, F., KIM, J., AND FELDMANN, A. Revisiting cacheability in times of user generated content. In *Proceedings of the 13th IEEE Global Internet Symposium* (2010).

KIM, J., SCHNEIDER, F., AGER, B., AND FELDMANN, A. Today's Usenet Usage: Characterizing NNTP Traffic. In *Proceedings of the 13th IEEE Global Internet Symposium* (2010).

POESE, I., FRANK, B., AGER, B., SMARAGDAKIS, G., AND FELDMANN, A. Improving Content Delivery using Provider-aided Distance Information. In *Proceedings of the ACM Internet Measurement Conference* (2010).

Under submission

Parts of this thesis are based on the following paper that is currently under submission.

Conferences

AGER, B., MÜHLBAUER, W., SMARAGDAKIS, G., AND UHLIG, S. Web content cartography. Under submission.

1 Introduction

The increasing number of users as well as their demand for more and richer content has led to an exponential growth of Internet traffic for more than 15 years [11, 30, 49, 67]. In addition, new applications and use cases have changed the type of traffic. For example, social networking enables users to publish their own content. This *user generated content* (UGC) is often published on popular sites such as YouTube, Twitter, and Facebook. Another reason are the offerings of interactive and multi-media content by content providers, e. g., Google Maps or IPTV services. Since 1998 peer-to-peer (P2P) protocols caused an even more radical change as they allow users to directly exchange large amounts of content: The peers transfer data directly without the need for an intermediary and often centralized server. However, as shown by recent studies [11,49,56] traffic is again dominated by HTTP, mostly at the expense of P2P.

This traffic growth increases the demands on the infrastructure components that form the Internet. The byte distribution per HTTP domain matches Zipf's law [56]. This means that most of the traffic is generated by few very popular services. The enormous demand for such popular content cannot be satisfied by the traditional hosting model in in which content is located on a single server. Instead, content providers need to scale up their delivery infrastructure, e. g., by using replication in large data centers or by buying service from content delivery infrastructures, e. g., Akamai or Limelight. Indeed the top 5 content infrastructures are responsible for more than 50 % of the HTTP traffic at a vantage point, covering 20.000 residential users at a large European Internet service provider (ISP). But not only content providers have to cope with the demand. The network infrastructure also needs to be constantly upgraded to keep up with the growing demand for content.

Any content delivery infrastructure—client-server or peer-to-peer based—consists of three main building blocks: *(i)* a number of servers from which content is served, *(ii)* a replication scheme that defines how content is replicated across the servers, and *(iii)* a mechanism to direct clients to these servers. In addition, a content delivery infrastructure can implement mechanisms to ensure freshness and availability of the content. A typical choice for a content delivery infrastructure is: *(i)* HTTP caches located in data centers for content serving, and *(ii)* a server assignment mechanism implemented via *(iii)* DNS to direct the user to the content cache. The replication of content is triggered implicitely by cache misses. This kind of service is often called content distribution network (CDN). In P2P based systems *(i)* , *(ii)* content replication is done on the peers themselves, and, *(iii)* e. g.,

a distributed hash table is utilized for directing peers to the content.

This describes the architecture of a content delivery infrastructure. However, there are more parties involved to deliver the content to the user:

1. Many content delivery infrastructures operate their servers at several geographic locations and/or have network connectivity to several ISPs. This location diversity—geographic location or network location—can be utilized in various ways by the content delivery infrastructure: to improve the overall performance of content delivery by choosing nearby content servers, to increase availability, to lower the cost for electricity [74] or bandwith, to accomodate flash crowds [9, 39, 99], etc.

2. ISPs can avoid network bottlenecks not only by simply adding more bandwith, but also by traffic engineering [21, 26, 28, 73] or the installation of network caches (cf. Chapter 3).

3. The user also influences the traffic he generates, by selecting the content he views and the applications he uses for downloading. There are more subtle influence factors, e. g., a user can use a third-party DNS resolver (cf. Chapter 4), virtual private networks, onion routing, or v4-to-v6 tunnels.

This means, the way that content is delivered in todays Internet is sculpted by at least three parties: the content delivery infrastructure, the Internet service providers (ISPs) on the network path, and the client itself.

In this thesis we characterize the impact of content delivery on the network: the transport of the content from the server location to the client location. In general, the client location is fixed—Web surfers typically do not change their physical location or their network connection to fetch content. However, the content location can be influenced: A content delivery infrastructure can select the content server, or an ISP can install a cache. Accordingly we study three aspects of content delivery: *(i)* how effective can caching be, *(ii)* what is the role of the server assignment mechanism and how does it interact with the user, and *(iii)* where from can content actually be fetched.

We utilize data sets from both active and passive measurements. This allows us to cover a wide range of abstraction levels from a detailed protocol level view of several content delivery mechanisms to the high-level picture of identifying and mapping the content infrastructures that are responsible for the most popular content.

1.1 Contributions

To understand how content delivery in today's Internet works we study how the content delivery infrastructures interact with ISPs as well as users.

1.1.1 Bringing Content Nearer to the User via Caching

One way for an ISP to increase the network performance while minimizing the overall traffic volume in its network is to reduce the number of hops the traffic has to be transported. An ISP can achieve this by moving content nearer to the consumer, e. g., by using caches. We examine the potential of caching HTTP, BitTorrent, eDonkey, and NNTP. These are the four most prevalent protocols and cover more than 70 % of the traffic volume at a vantage point within a large European ISP.

We find that BitTorrent and eDonkey are very well suited for caching when using the appropriate mechanisms. However, they only contribute a small fraction of the overall traffic volume. HTTP is responsible for the largest traffic share but is quite limited in its cacheability. NNTP is hardly cacheable at all.

1.1.2 The Role of DNS in Selecting Servers

Highly distributed and well connected content infrastructures such as CDNs or cloud services contribute a large fraction of the HTTP traffic. This leads us to further investigate the impact of their specific architecture on content delivery. Many content infrastructures, e. g., those owned by Akamai, Google, Limelight, Edgecast, Amazon, Footprint, and Rackspace, rely on DNS to assign content servers to clients. For performance reasons the client location is often used in the content server selection process. However, due to the design of DNS, the best achievable approximation of the client location is the location of the DNS resolver. Thus, we study the behaviour of DNS in over 50 commercial ISPs worldwide and compare them with widely-used third-party resolvers, GoogleDNS and OpenDNS.

We find that third-party resolvers, in general, do not manage to direct clients to content caches available inside the client's ISP, contrary to the ISP's DNS resolvers. Moreover, we find that the users of a surprisingly large number of ISPs suffer from poor latency towards their DNS resolvers, and that the DNS setup of serveral ISPs shows evidence of load balancing which results in reduced DNS cache utilization.

1.1.3 Web Content Cartography

Next we ask the question: From where is content in principle available? To answer we propose *Web content cartography*: building an abstract representation of the Web hosting infrastructure at multiple levels such as ASs, prefixes, and geographic locations.

We use DNS observations in 78 autonomous systems (ASs) to estimate the location diversity of popular hostnames. Using these we build maps of hosting infrastructures and highlight the prevalence of content replication. Next, we develop a lightweight technique to identify individual infrastructures based on their

network footprint. This allows us to quantify how much content is available at a given location, e. g., an AS or a country. Finally we define an index that reflects the degree of exclusively hosted content within an organization.

1.1.4 Discussion

To put our findings in context we complement them with results of contemporary work. We argue that client-server based hosting will stay the most important source of content for at least the next few years. We review general design criteria for content delivery infrastructures and ask if energy consumption should become a more central design objective. We give recommendations to ISPs, content delivery infrastructures, and users based on our findings, and conclude with a list of open questions.

1.2 Structure of the Thesis

The rest of the thesis is structured as follows: In Chapter 2 we briefly discuss what Internet traffic looks like today, show how content delivery infrastructures work, and give a short overview of the prevalent protocols used for content delivery. In Chapter 3 we analyze the potential of caching content, and examine the impact of DNS server location on server selection in Chapter 4. In Chapter 5 we map content to locations from where it can be served. We discuss the results in Chapter 6, and conclude in Chapter 7.

2 Background

In this chapter we review how Internet traffic developed and how it is composed today. In addition we briefly introduce the technical background of DNS and the four most prevalent protocols for content delivery in the Internet today.

2.1 Internet Traffic Today

In this section we briefly discuss recent findings in Internet traffic characterization, and give background information on the design of content infrastructures.

2.1.1 Application Volume and Traffic Mix

Internet traffic growth has been shown to be largely consistent with an exponential growth model [11, 30, 67] and recent studies estimate a annual increase of Internet traffic in the order of 30 % to 45 % [11, 30, 49]. However, the composition of Internet traffic is subject to change with the development of new applications and use cases. A number of studies has been published in the last four years reporting on the traffic shares of popular protocols in the Internet [11, 19, 30, 49, 56, 79–82]. Several studies found that P2P was the dominating protocol before 2009, with 65 % to 83 % of the total traffic volume [30, 69, 81, 82]. In strong contrast to this recent observerations show that HTTP has become the transfer protocol of choice for many of today's Internet applications. Indeed, with 52 % to 65 % of all traffic volume [19, 30, 49, 56, 79, 80] HTTP has again become the dominating protocol in today's Internet, while P2P usage has dropped to between 12.3 % and 23 %[1]. The reason for this shift are high-volume applications such as file-sharing and video streaming that use HTTP instead of specialized protocols such as FTP [72], RTP [83], RTSP [84], or P2P protocols as mentioned above [19, 49, 56]. In addition, NNTP [22] has been observed to be responsible for a considerable fraction of the traffic [19, 49, 56]. We show that NNTP is used today as a file-sharing platform as well [43].

[1] Sandvine [80] reports up to 39 % of the total bytes being P2P in Asia

2.1.2 A Shift in the Internet Topology

Based on flow data covering inter-AS traffic Labowitz et al. observe content consolidation resulting in a shift in the Internet topology [49]. According to Labovitz et al. large content contributors coined "hyper-giants" can be found amongst the top ASs in terms of inter-AS traffic volume. These hyper-giants typically have direct peerings with ISPs, therefore the Internet structure is not as strongly hierarchical as it used to be. Moreover, the shift induced by content consolidation can also be observed in the traffic volume: in 2007, the top 150 ASs contributed 30 % of the traffic, in 2009 they contribute more than 50 % [49]. In Chapter 5 we explain this shift with the power of content.

2.1.3 Content Infrastructures

The sheer demand for certain types of content requires scalable delivery systems. One scalable way of distributing content are P2P protocols, which use the upload capacity of downloaders to cope with the demand. Client-server based approaches achieve scalability by server-side replication of the content, thus multiplying the available resources for content delivery. Both approaches are used successfully today, yet both have their advantages and disadvantages: For example, content delivery networks (CDNs) utilizing P2P protocols benefit from the free (as in cost) upload bandwith provided by the peers, yet often suffer from the asymmetricity in available bandwith of end customers [43], cf. Chapter 3. With client-server protocols the content provider can buy as much upload capacity as needed at the tradeoff of having to pay for the ressources. Leighton discusses [53] some of the various tradeoffs. Another dimension is introduced to the problem if considerable computing ressources are required to generate content specific to a user's request, e. g., on dynamic Web sites such as Amazon's Web shop or Google search. In this case the software has to be replicated, not only the content. This kind of service can be built on top of a *cloud* service.

An example for P2P supported content distribution is Blizzard Entertainment, who is distributing several of their games and their patches with the help of BitTorrent[2] and HTTP in a hybrid approach. Other examples include companies such as Akamai, Amazon, Google, Edgecast, Footprint, Limelight, and Microsoft who own CDNs and/or clouds. They utilize them for offering their own content (e. g., Amazon), for implementing services on top of the cloud (e. g., Google), and for selling the CDN service as a product to content providers (e. g., Akamai).

To build a client-server based content infrastructure three main components are needed: *(i)* a large number of servers from which the content can be served, *(ii)* a mechanism to direct clients to one (or a subset) of these servers, and *(iii)* a replication scheme that defines how content is replicated across the servers. In addition,

[2] `http://www.wowpedia.org/Blizzard_Downloader`

mechanisms to guarantee the freshness and availability of the replicated content can be implemented. Content infrastructures can evaluate multiple dimensions for server selection, e. g., the load on the server [9, 39, 99], energy [74] and bandwith costs, the location of the client [53, 90], and the freshness of the cache.

As indicated by the overall dominance of HTTP in todays traffic HTTP based content infrastructures are the most prevalent. In this case the content replication servers are typically caching proxies or cloud servers. As a direction mechanism both HTTP and DNS can be used. Typically DNS is chosen to save the extra round trip times of an additional HTTP connection. However, for embedded objects and for bulk hosting services an HTTP based redirection mechanism is also feasible. For example, according to our own investigations in YouTube the video server selection happens both with the help of HTTP and DNS, and for RapidShare the selection options of the content server are embedded into one of the pages presented before the actual bulk download is initiated.

2.2 Protocols

In this section we give a short overview of the technical details of the protocols examined in this thesis. We start by explaining DNS, one of the fundamental protocols in the Internet, which is also used to implement load balancing in CDNs. Familiarity with DNS is necessary to understand Chapter 4 and Chapter 5. We continue with the most prevalent client-server based protocols in our traffic mix, HTTP and NNTP, and the most volumnious P2P protocols, BitTorrent and eDonkey. A basic familiarity with these protocols is helpful to understand the caching analysis in Chapter 3.

2.2.1 Domain Name System (DNS)

When the domain name system (DNS) was introduced in 1983, its purpose was to resolve host names into IP addresses in a more scalable fashion than the until then used `hosts` file. DNS relies on a distributed database with a hierarchical structure. The root zone of the DNS system is centrally administered and serves its *zone* information via a collection of *root servers*. The root servers delegate responsibility for specific parts (*domains*) of the hierarchy to other *name servers*, which may in turn delegate the responsibility to other name servers. At the end, each site is responsible for its own zone and maintains its own database containing its information and operates an *authoritative* name server. An alternative view of this name space is one of a tree with labels at the nodes separated by dots. Information associated with any particular name is composed of *resource records* (RRs) which contain the class and type of the resource and data describing it. Multiple RRs with the same name, type and class are called a resource record set (RRset).

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55911
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 9, ADDITIONAL: 8

;; QUESTION SECTION:
;www.audi.de.                    IN  A

;; ANSWER SECTION:
www.audi.de.               575    IN  CNAME   www.audi.de.edgesuite.net.
www.audi.de.edgesuite.net. 21275  IN  CNAME   a1845.ga.akamai.net.
a1845.ga.akamai.net.       20     IN  A       192.168.254.1
a1845.ga.akamai.net.       20     IN  A       192.168.254.2

;; AUTHORITY SECTION:
ga.akamai.net.             1475   IN  NS      n0ga.akamai.net.
[...]

;; ADDITIONAL SECTION:
n0ga.akamai.net.           667    IN  A       192.168.255.1
[...]
```

Figure 2.1: An example showing how the DNS reply for content hosted on a content infrastructure could look like. The format resembles the output of the `dig` command line tool.

The whole database is usually queried by an end-host's *stub resolver* using a local name server called *caching resolver*. If this name server receives a query for information that it does not have, it must fetch this information from another name server. If the server does not know how to contact the authoritative server for a zone, it will query a root server[3]. The root server will *refer* the resolver to another server that is authoritative for the domain that is immediately below the root and of which the zone is a part. The resolver will then query this server, and so forth, stepping down the tree from the root to the desired zone.

For efficiency reasons DNS relies heavily on caching [40]. All information that a name server delivers to a resolver can be cached for a duration specified in the TTL field of the *resource records* (RR). Caching today is commonly also performed on end-hosts by the operating system's *stub resolver*, as well as applications, e. g., Web browsers.

Figure 2.1 shows an example DNS reply as it could be returend by a resolver when querying a hostname served by a content infrastructure. In this case, the answer section of the reply contains a chain of `CNAME` records, i. e., name redirections.

[3]The first query can go to some authoritative server below the root if there exists cached information.

```
GET / HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (...)
Accept: text/html,application/xhtml+xml, ...
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Cache-Control: max-age=0
```

Figure 2.2: Example HTTP request

Eventually the `CNAME` chain points to an `A` ressource record set (RRset) containing two IP addresses, from which the Web site content can be fetched.

2.2.2 Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol (HTTP) has been developed as the protcol underlying the World Wide Web [4]. HTTP was initially designed as a dedicated protocol for information research [5]. Having scalability in mind a stateless design was chosen [4]. The most recent incarnation of HTTP is version 1.1 and is standardized by the IETF in RFC 2616 [27].

HTTP communication works by request-response pairs. Every HTTP message consists of an introductory line, a set of HTTP *headers* refining the meaning of the message and an optional HTTP *body* containing the actual data.

The introductory line in an HTTP request (Figure 2.2) consists of a *method*, a server-side *path*, and the HTTP version in use. The introductory line in an HTTP response (Figure 2.3) starts out with the HTTP version in use, followed by a standardized three-digit status code and a textual status description. The status code tells the requester about the success of the query or indicates the reason of an error.

HTTP Headers

HTTP headers allow for the easy extensibilty of HTTP messages and serve a number of different purposes, e. g., the specification of the queried hostname to allow virtual hosting, the transport of meta information, the control of persistent connections, and cache control. HTTP headers consist of a field name followed by a colon and the field value, cf. Figures 2.2 and 2.3. The `Host` header is mandatory for requests since HTTP 1.1. The meta information encompasses information about the file type, the character set in use, preferred languages, compression,

```
HTTP/1.1 200 OK
Server: Apache
Content-Type: text/html; charset=iso-8859-1
Date: Fri, 18 Feb 2011 17:06:23 GMT
Keep-Alive: timeout=20, max=966
Expires: Fri, 18 Feb 2012 17:06:23 GMT
Content-Length: 232
Connection: Keep-Alive

<html><head><title>Example pape</title></head>
<body>...
```

Figure 2.3: Example HTTP reply

and length information. Moreover HTTP headers can be used for stateful session management. To this end, HTTP allows a server to set a "cookie" [47, 48] in a response. This cookie is chosen uniquely by the server on a per client basis and will be sent by the client alongside all subsequent HTTP requests to the same domain. This allows the server to track all requests initiated by a client and therefore allows to personalize the content.

Cache-control in HTTP

HTTP cache control is a mechansim that allows fine grained control over the re-use of saved copies of a particular download. In HTTP both the browser on the end host as well as dedictated nodes in the network, HTTP proxies, can implement an HTTP cache. Cache control has evolved over different iterations of the HTTP protocol, with newer version of HTTP re-implementing functionality of older ones in different headers, thus making the cache decision process quite complex. The headers relevant for caching are `Pragma`, `Cache-Control`, `Expires`, `Last-Modified`, `Etag`, `Date`, and `Authorization` [27].

It is noteworthy that HTTP cookies [47, 48] do not per-se invalidate the cacheability of an HTTP response. Instead the server has to explicitly specify the caching policy. For example, this enables the server to allow caching the body of a response and require an update for only the `Set-Cookie/Set-Cookie2` header on every subsequent hit to the HTTP cache.

2.2.3 NNTP

The Network News Transfer Protocol (NNTP) is the protocol underlying the Usenet. The Usenet is one of the oldest parts of the Internet. In fact, it predates the world wide Web by more than 10 years. It was designed and is still being used as a system to exchange messages (called articles) between groups of users

with similar interests—a functionality that today is also provided by, e. g., Web forums.

The Usenet is structured in a (pseudo) hierarchy of groups, e. g., `comp.os.minix` or `comp.os.linux.networking`. Anyone with access to a news server can subscribe to any of the news groups and then read or post messages within the group. The news servers are usually hosted by ISPs or at universities[4]. They are connected to form an overlay network, which is used to replicate articles between servers so that everyone has access to all articles not just those posted to the local news server. However, there is one limitation: It is the decision of the administrator of a news server if a particular news group is hosted on his server, e. g., nowadays most news servers typically do not host the majority of the binary groups in order to avoid the risk of excessive bandwidth usage. NNTP is used for both client-server as well as server-server communication.

The Network News Transfer Protocol (RFC 3977 [22]) and its message formats (RFCs 5536, 5537, until Nov 2009: RFC 1036) are very similar to SMTP. In fact, many of the article header fields are identical to the email message header fields, e. g., `From` and `Subject`. In addition, there are NNTP specific headers, for example the `Message-ID` header, which contains a unique identifier for an article valid on all news servers.

When a client connetct to a NNTP server, the server answers with a greeting message. Then the client can issue its commands. The server replies to each client command with a three digit status code, a status message, and an optional data block in "multi-line" encoding which contains, e. g., the requested article. Likewise, a client can send a "multi-line" data block to the server, e. g., to post an article. At the beginning of the connection a news server may require authorization before the client can issue any commands. Examples commands include selecting a group (`GROUP`), listing the articles within a group (`LISTGROUP` or (`X`)`OVER`), and fetching (`ARTICLE`, `BODY` and `HEAD`) or sending (`POST`) articles.

2.2.4 BitTorrent

BitTorrent is a peer-to-peer (P2P) protocol optimized for the fast distribution of potentially large files to many hosts. BitTorrent achieves this by splitting a file in *pieces* that are redistributed by a peer as soon as it has received a full copy of the piece and well-chosen heuristics [52] for optimizing the replication. BitTorrent has been developed by Bram Cohen since 2001[5] and has since been standardized in the form of "BitTorrent Enhancement Proposals" (BEPs) [33].

BEP 3 [33] describes the main protocol and how the different entities in in BitTorrent *swarm* interact with each other. To start a new swarm the *seeder* publishes a

[4] Users whose ISP does not offer access to a news server can subscribe for a small fee to independent servers, e. g., `http://news.individual.net/`

[5] `http://finance.groups.yahoo.com/group/decentralization/message/3160`

metainfo file of the content on a Webserver. The metainfo file describes the content and the location of a *tracker* that coordinates the swarm. In addition the seeder starts a BitTorrent client with the metainfo file and a full copy of the content. This client is called the *initial seed*. If a *peer* wants to participate in a download, it uses the metainfo to connect to the tracker. The tracker informs the peer about other participants in the swarm, so that the peer can contact them and start downloading *pieces* of the content. A piece has a typical size of about 1 MB. As soon as a peer has received a full copy of a piece it will verify it with an SHA-1 checksum from the metafile and start offering it to the rest of the swarm. After the content is fully fetched a peer becomes a *seed* and can continue uploading the content.

The BitTorrent Tracker

The BitTorrent tracker is either an HTTP server according to BEP 3 [33] or a Kademlia based [59] distributed hash table (DHT) according to BEP 5 [33]. In addition, at least one incompatible version of a distributed hash table approach has been implemented in Vuze[6]. The main function of the tracker is to hand out lists of participants of a swarm to peers to allow building the P2P overlay.

To further reduce load on the often centralized tracker various flavours of *peer exchange protocols* (PEX) have been proposed and implemented. The basic idea is that the tracker is mainly used to bootstrap a client into a swarm. From then on, the peer can trade information about its peers with its direct neighbours to quickly gain a large list of potential peers while still maintaining scalability.

BitTorrent connections

BitTorrent connections between two peers are initiated with a handshake. Among other things, the handshake identifies the torrent that is handled over this connection. It also serves to inform the opposite side about extended capabilities [6, 62]. In case both clients are capable of the Azureus Messaging Protocol [6] a direct switch to AZMP is performed. After this handshake, the message exchange starts.

BitTorrent protocol messages

A BitTorrent message is composed of a length header followed by a message type and the actual message payload. The BitTorrent protocol specification (BEP 3 [33]) introduces nine message types, and a zero-length keep-alive pseudo message. Of those nine messages four are used to implement the tit-for-tat algoritm [52], two are used to inform peers about the content a client already has downloaded, and three are used to handle the actual exchange of content.

In particular, the `have` and `bitfield` messages are used to inform a peer about available content. The `bitfield` message is sent directly after a handshake and

[6] `http://wiki.vuze.com/w/Distributed_hash_table`

gives an overview of available pieces on the sending peer. In contrast, a `have` message is sent to inform a peer when a new piece becomes available later in the connection. The `request` and `piece` messages are used to request and transfer content. A peculiarity in the BitTorrent protocol is that the `request` and corresponding `piece` message are only for a *chunk*, i. e., typically a 16 kB part of a piece, and not the full piece.

The standard message types can be complemented by protocol extensions. Most prevalent in the traffic we looked at are the LibTorrent Extension Protocol [62] (LTEP) and the Azureus Messaging Protocol [6] (AZMP). LTEP is integrated into the standard protocol by embedding all LTEP messages into `extended` messages. AZMP defines an entirely different self-contained protocol thus reimplementing the standard message types. Both allow the specification of arbitrary strings to denote messages types. Examples for protocol extensions are peer exchange (PEX) messages as described earlier in this section, chat messages, and transfer of metainfo files.

2.2.5 eDonkey

EDoneky is hybrid peer-to-peer network used for file-sharing. It consists of a server software that is used for indexing files and searching content, and a client part that is used for downloading and uploading [34]. The server part uses a gossip protocol to exchange information about currently active servers. This information can be aquired by clients upon connecting to a server to maintain a local list of active servers [34]. Newer versions of eDonkey software make use of Kad—a distributed hash table based on the Kademlia algorithm [59]—for searching.

The file exchange happens directly between clients. Similarly to BitTorrent, the file verification is performed with checksums over parts of files. In eDonkey, these parts are of 9.28 MB size[7]. The actual download happens in smaller parts of approximately 180 kB size [42], similar to BitTorrent chunks.

2.3 Summary

To summarize: Today's Internet traffic mix is dominated by HTTP traffic. The underlying reason is the shift of many volumnious applications towards HTTP as the delivery protocol. To cope with the demand for certain popular contents the introduction of content delivery networks is necessary. These networks can be built on P2P protocols or on client-server protocols. A popular protocol choice for CDNs is HTTP, and DNS is used as the load balancing mechanism.

[7] `http://www.emule-project.net/home/perl/help.cgi?l=1&rm=show_topic&topic_id=232`

3 On the Effect(iveness) of Caching

We start our investigation by examining the low-level behaviour of the most prominent protocols at a vantage point of a large European ISP covering about 20,000 residential users. To evaluate an ISP's opportunities to optimize its network, we estimate the potential of caching HTTP, BitTorrent, eDonkey, and NNTP.

3.1 Introduction

The Internet has evolved into a system where users can easily share content with their friends and/or other users via applications such as wikis, blogs, online social networks, P2P file-sharing applications, One-Click Hosters, or video portals, to name a few of the most well-known user generated content (UGC) services. In terms of volume, multi-media content including photos, music, and videos, as well as software downloads and updates, are major contributors and together responsible for most of the Internet traffic [49, 56, 79, 82]. Indeed, HTTP is again accounting for more than 50 % of the traffic [49, 56, 79, 82] and is hardly (mis-)used as transport protocol for other applications [56]. Among the causes for the increase of HTTP traffic are One-Click-Hosters such as `rapidshare.com` or `uploaded.to` and the increase of streaming content, e. g., offered by `youtube.com`.

In the early stages of the Internet Web caches were very popular. However, the efficiency of Web caches [23, 75] decreased drastically as the popularity of advanced features increased: dynamic/personalized Web pages (via cookies), AJAX-based applications, etc. Reexamining today's content we find that a large fraction, especially multi-media content, is static and therefore it might be rewarding to re-evaluate the caching potential. This is confirmed by recent caching efficiency studies for specific applications, e. g., Gnutella [35], Fasttrack [100], YouTube [103], BitTorrent [41], and Web [19]. Rather than focusing on a specific application, we in this chapter study a *set of application protocols*. For these we investigate the potential of caches, traffic redirection for sharing content, as well as causes of non-cacheability.

In this chapter we present observations based on passive packet-level monitoring of more than 20,000 residential DSL lines from a major European ISP. This unique vantage point coupled with our application protocol analysis capabilities enables a more comprehensive and detailed characterizations than previously possible. We focus on the cacheability of multiple applications that are predom-

Table 3.1: Overview of anonymized packet traces and summaries.

			Application Volume			
Name	Start date	Dur	Size	HTTP	BT	eDonkey	NNTP
APR09	Wed 01 Apr'09 2am	24 h	>4 TB	58 %	9 %	3 %	2 %
AUG09	Fri 21 Aug'09 2am	48 h	>11 TB	63 %	9 %	3 %	2 %
HTTP-14d	Wed 09 Sep'09 3am	14 d	> 0.8 TB	corresponds to > 40 TB HTTP			
NNTP-15d	Wed 05 Aug'09 3am	15 d	> 2 GB	corresponds to > 2 TB NNTP			
BitTorrent-14d	Sat 20 Jun'09 3am	14 d	> 80 GB	corresponds to > 5 TB BitTorrent			

inantly used for sharing content in our environment[1]: *(i)* HTTP, *(ii)* BitTorrent, *(iii)* eDonkey, and *(iv)* NNTP. Note, that HTTP and NNTP are client/server protocols while BitTorrent and eDonkey are P2P protocols. It might seem odd to include NNTP, the protocol used by Usenet, but we surprisingly find that NNTP accounts for more than 2 % of the total traffic volume and seems to be used as an alternative for file-sharing [43]. In previous work, Karagiannis et al. [41] study the cacheability of BitTorrent before it became one of the dominant file-sharing applications based on data from a residential university enviroment. Erman et al. [19] only focus on the cacheability of Web traffic.

We find that the story for caching is ambivalent. For client/server-based applications, including NNTP and some Web domain classes, e. g., One Click Hosters, caching is ineffective. For other HTTP services, e. g., Software/Updates, we observe caching efficiencies up to 90 %. In addtion, for some domains, using opportunistic cache heuristics improves cacheability substantially. For P2P protocols, especially BitTorrent, there is substantial potential for caching if the cache actively participates in the protocol. Moreover, traffic localization via mechanisms, such as those proposed by, e. g., Aggarwal et al. [1], Xie et al [101], Choffnes and Bustamante [10], and currently under discussion within the IETF ALTO [85] working group, are promising[2]. Traffic localization in effect uses local peers as cache.

3.2 Data Sets

We base our study on multiple sets of anonymized packet-level observations of residential DSL connections collected at aggregation points within a large European ISP. Our monitor, using Endace monitoring cards, allows us to observe the traffic of more than 20,000 DSL lines to the Internet. The data anonymization, classification, as well as application protocol specific header extraction is performed immediately on the secured measurement infrastructure using the Bro NIDS [68]

[1] In previous work Maier et al. [56] found that at our vantage point HTTP is responsible for more than 50 % of the traffic while P2P (BitTorrent and eDonkey) contribute less than 15 %. Even assuming all unclassified traffic to be P2P in total it only accounts for less than 30 %.

[2] The key idea is that ISPs and P2P users collaborate to locate close by peers.

with dynamic protocol detection (DPD) [18]. For HTTP we extend the standard protocol analyzer to compute MD5 hashes across the HTTP bodies. In addition, we developed DPD-based analyzers for NNTP, eDonkey, and BitTorrent [33], including the Azureus Messaging Protocol [6] and the LibTorrent Extension Protocol.

We use an anonymized 24 h packet trace collected in April 2009 (APR09) and an anonymized 48 h trace collected in August 2009 (AUG09). APR09 is the same data set as analyzed by Maier et al. [56]. While we typically do not experience any packet loss, there are several multi-second periods (less than 5 minutes overall per packet trace) with no packets due to OS/file-system interactions.

For studying long term effects of cacheability we used Bro's online analysis capabilities to collect several anonymized protocol specific trace summaries (BitTorrent-14d, NNTP-15d, HTTP-14d) which span at least 2 weeks. Due to the amount of traffic at our vantage point and the resource intensive analysis we gather the online trace summaries one at a time. Table 3.1 summarizes characteristics of the traces, including their start, duration, size, and application mix.

With regards to the application mix, see Table 3.1, Maier et al. [56] find that HTTP, NNTP, BitTorrent, and eDonkey each contribute a significant amount of traffic. Moreover, their total traffic adds up to more than 72 % of the overall traffic at our vantage point. Similar protocol distributions have been observed at different times and at other locations of the same ISP.

Surprisingly, NNTP accounts for more than 2 % of the traffic. However, a detailed investigation [43] shows that 99 % of the current traffic is bound to/from fee-based NNTP servers. These fee-based offers are competing with One-Click-Hosters, as the client/server-based alternative for file-sharing.

3.3 Terminology and Approach

Before delving into the details of the application specific cacheability analysis we introduce our terminology. Caching refers to saving a copy of a reply to a request on a server—the *cache*—with the intention to satisfy subsequent requests for the same content from the cache instead of the origin. For us all requests to a content item except the first one are in principle *cacheable*. We, in this chapter, explore the potential for cacheability rather than focusing on specific caching heuristics, thus we are never limited by disk space.

We use the following two cacheability metrics: number of cacheable requests and cacheable volume. If k_i denotes the total number of downloads for item i of size s_i the cacheable volume of n items is computed as $cacheability$ according to Equation (3.1). The cacheable requests are calculated using the same equation with all sizes equal to 1.

$$cacheability = \frac{\sum_{i=1..n}(k_i - 1) \cdot s_i}{\sum_{i=1..n} k_i \cdot s_i} \tag{3.1}$$

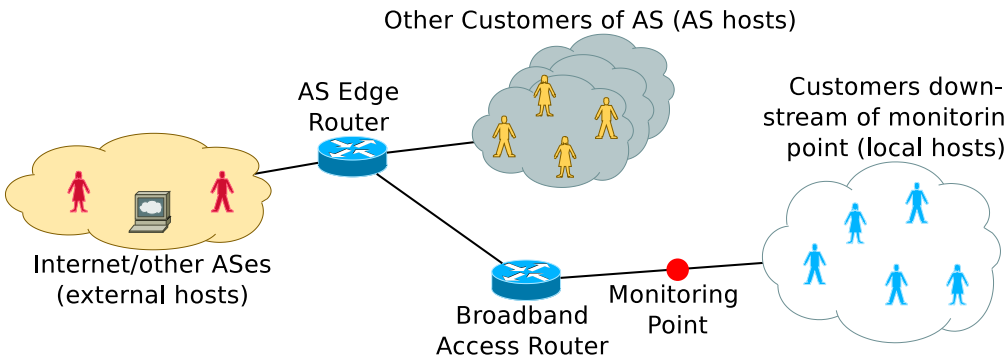


Figure 3.1: Vantage point and sets of hosts: *local*, *AS*, and *external*

To estimate the impact of caching it is important to consider which mechanisms are available to redirect requests to the cache and where the cache is located. If P2P content is available at multiple different locations one can use mechanisms [1,10,101] currently under discussion within the IETF ALTO [85] group. In addition, it may be beneficial or even necessary to setup dedicated caches within the network. With regards to network location, see Figure 3.1, we distinguish between *(i)* hosts that are downstream from the monitor, *local* hosts, *(ii)* hosts that are within the ISP's autonomous system (AS) yet not local, *AS* hosts, and *(iii) external* hosts.

3.3.1 Peer-to-Peer

In P2P the unit for caching is either the complete object or the transfer unit if the P2P protocol splits the file into chunks, e. g., for more efficient transfers. The former captures the number of users interested in an object. The latter corresponds to the observed traffic. Since users may not download complete objects while online these metrics can differ significantly. In BitTorrent the complete objects are the torrents and the transfer units are the (fixed size) blocks. In eDonkey the objects are the files and transfer units are blocks.

Given that a BitTorrent retrieval can last multiple days it is likely that a any trace includes some partial downloads. Regarding a cacheability analysis the lack of knowledge about transfers that occurred before the start of the observation period is problematic as these could have primed the cache. Fortunately, for BitTorrent we can leverage the *bitfield* and *have* messages to infer which transfers occurred prior to the trace start using a similar methodology as Karagiannis et al. [41].

Given that P2P clients are also P2P servers we can in addition estimate a lower bound for the cacheability for AS hosts and external hosts. Thus, we can study the potential of P2P cacheability with regards to *(i)* the number of peers interested in an object, referred to as *peers*, *(ii)* transferred blocks, called *blocks*, and *(iii)* transferred blocks given a primed cache, called *primed*, for all three classes of

hosts. To prime the cache in the third scenario, we make all blocks available in our cache that are announced in `bitfield` and `have` messages by the peers within the respective host sets.

3.3.2 Client-Server

For HTTP and NNTP we note the following differences: *(i)* Each expression of interest for an object corresponds to a full or partial download of the whole object; *(ii)* we cannot infer information for clients outside of our local network. Moreover, cache control in HTTP 1.1 offers a lot of options via explicit cache-control headers. To identify HTTP objects we use both—the size and the MD5 sum of the HTTP body (*object ID*).

With regards to cacheability the units of interest for NNTP are articles. NNTP articles are comparable to objects in HTTP or emails in IMAP. Commands for accessing articles are `ARTICLE`, `HEAD`, and `BODY`, which request to download the whole article, only the headers, and only the body, respectively. Although NNTP includes a large number of commands the above ones are the only ones not used for navigating, controlling, and listing. Accordingly, the other commands contribute only a tiny fraction (less than 2 ‰) to the overall NNTP volume. NNTP differs from HTTP as article IDs are unique and articles cannot be modified except by the administrator. To identify objects we thus use the article ID.

We study a range of different cacheability scenarios ranging from *ideal* to *realistic*, see Table 3.2. The *ideal* scenario is used to derive an upper bound on the cacheability using Equation (3.1). This is the only scenario applicable to NNTP. However, for HTTP this scenario is practically infeasible as one would have to reliable predict that two requests (different URLs) are for the same content. We observe that CDNs are often performing load balancing using one of two possible mechanisms: Either load balancing is performed during the DNS resolution step and different IP addresses are returned to different resolvers. Or load-balancing is implemented inside the HTTP protocol. In this case the same content is offered under URLs that only differ in the host name. Moreover, a realistic cache is typically unable to cache objects if two clients use different query methods or if the return code differs. The *domain* scenario thus adds these three criteria to the object ID, and therefore gives us insight on the impact of load balancing over different servers. To account for the observation that identical object IDs can be hosted by the same providers at different URLs (host name and/or path), e. g., to help with load balancing or to accommodate GET parameters we use the scenario *complete*.

In the fourth scenario, *full*, we simulate an actual HTTP cache which respects the cache control headers: `Pragma`, `Cache-Control`, `Expires`, `Last-Modified`, `Etag`, `Authorization`, and the HTTP methods as specified in RFC 2616. However, unlike a real cache, we are still using the object ID. We use this scenario only to evaluate the negative impact of the object ID on cacheability.

Table 3.2: Criteria for identifying cacheable objects in HTTP.

scenario	object ID	HTTP method	return code	2nd level domain	host	path	cache control
ideal	✔						
domain	✔	✔	✔	✔			
complete	✔	✔	✔		✔	✔	
full	✔	✔	✔		✔	✔	✔
realistic		✔	✔		✔	✔	✔

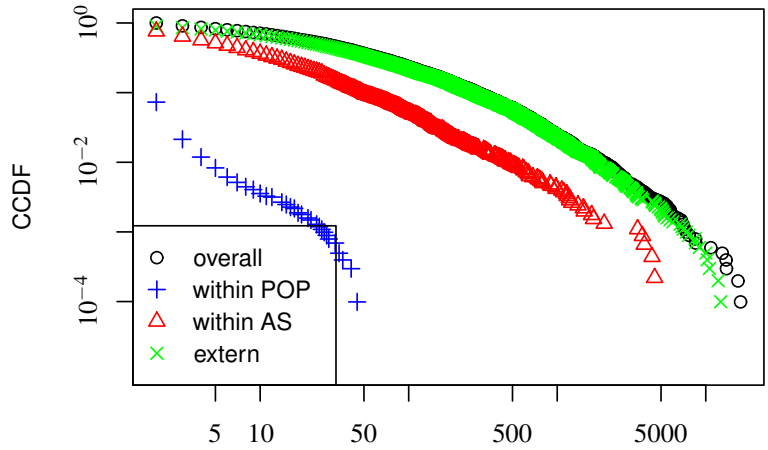


Figure 3.2: Complementary cumulative distribution function (CCDF) of BitTorrent analysis for BitTorrent-14d: Peers per torrent.

In the fifth and last scenario, *realistic*, the cache behaves like an ideal caching proxy server with unlimited disk space. The cache always performs the best possible caching option. For example, if an object is stale it is not purged from the cache. Rather the next query for the same object causes a conditional request to the HTTP server, thus allowing the refresh of a cached object without actually downloading it. If an object is already partially cached when a request occurs, only the missing parts are fetched from the server if the object has not changed in the meantime. If no expiration time is set we use a heuristic motivated by RFC 2616: $t_{expiry} = t_{now} + min(0.1 \cdot (t_{now} - t_{Last\text{-}Modified}), 1\,\text{day})$. Table 3.2 summarizes the five HTTP scenarios. Note, only the scenarios *full* and *realistic* are using timeouts.

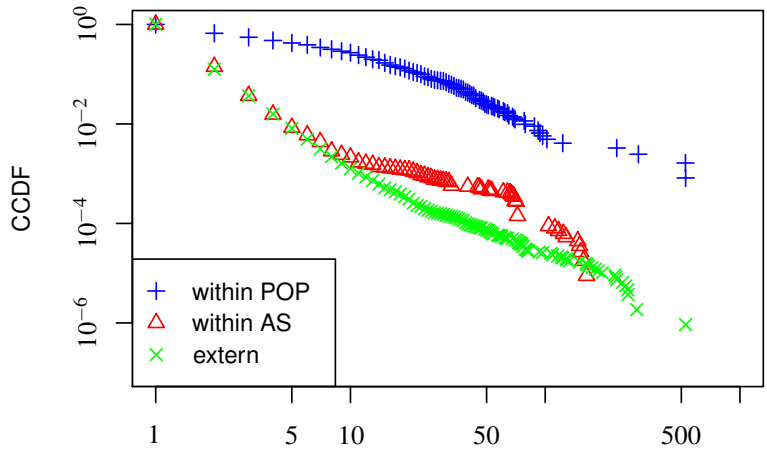


Figure 3.3: Complementary cumulative distribution functions (CCDF) of BitTorrent analysis for BitTorrent-14d: Torrents per peer.

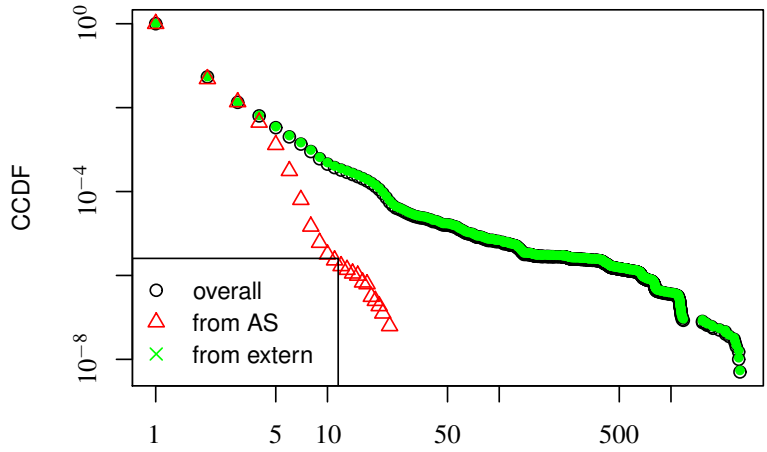


Figure 3.4: Complementary cumulative distribution functions (CCDF) of BitTorrent analysis for BitTorrent-14d: Downloads per block.

3.4 Results

Our analysis shows that the potential cacheability differs substantially among the application protocols.

3.4.1 Peer-to-Peer

Even though P2P is no longer dominating residential traffic, P2P still contributes a significant amount of traffic. In fact, BitTorrent and eDonkey are the second and third most voluminous protocols after HTTP, with more than 10 % of the total

traffic volume.

In Figure 3.2, Figure 3.3, and Figure 3.4, we show the complementary cumulative distribution functions of the distributions we use to calculate cacheability of BitTorrent for BitTorrent-14d. The first indicator for estimating the potential of caching is the number of peers per torrent swarm, scenario *peers*. Figure 3.2 shows this distribution for local, AS, external, and all hosts. We see that many torrents have a sizable number of peers within the AS.

If all peers within the AS would do a complete download 97 % of the bytes are downloadable from peers within the AS—corresponding to an AS-wide caching efficiency of 97 %. This is a lower bound as there may be other peers within the AS. When we consider only local hosts caching efficiency drops substantially to 27 %. Nevertheless, this is still promising for caching.

When focusing on scenario *primed*, we find that 85 % of the download volume are in principle cacheable. However, this cacheability result relies on the availability of hosts that are torrent seeders (learned via the bitfield and have messages). We find that some hosts are seeders for a large number of torrents, see Figure 3.3, and that these are online for substantial time periods. This is consistent with the results of Stutzbach and Rejaie [89]. However, if we only consider actual downloads, scenario *block*, the cacheability drops to roughly 9 %, see Figure 3.4, contrary to the previous results. Part of the reason is that we may not have observed a flashcrowd effect during the 14 day observation period. Another reason is that clients often are seeders for many torrents but are only actively involved in a small number of downloads. For data from 2004 from a residential complex at a university Karagiannis et al [41] found that the cacheability for actual downloads is between 6 and 11.8 % which is consistent with our results. However, they saw a substantially lower overall cache efficiency with less than 18.5 %.

We also note that almost all (> 89 %) of the chunks are downloaded from and uploaded to external hosts even though the content is available locally, see Figure 3.4. We are able to identify such content by inspecting the bitfield messages of the peers. This shows the potential of both P2P neighbor selection strategies [1, 10, 101] as well as caches and confirms the results of Plissonneau et al. [70] for eDonkey and Karagiannis et al. [41].

To check if our observations are biased by the language of the content we examine the geolocation of the BitTorrent traffic with the help of the ISP. Figure 3.5 shows the amount of BitTorrent traffic downloaded from or uploaded to different locations normalized by the *overall* BitTorrent traffic volume. We distinguish between continents and within Europe we separate the local language region of the vantage point from the rest of Europe. Only one third of the content is being downloaded from the same language region. This indicates that while there many be some bias due to language it is not the dominating effect with regards to cacheability. The overall download volume is roughly twice that of the upload volume. This effect may be due to the asymmetry of the underlying DSL lines. In addition, Europe seems to be preferable as the upload to download ratio is better.

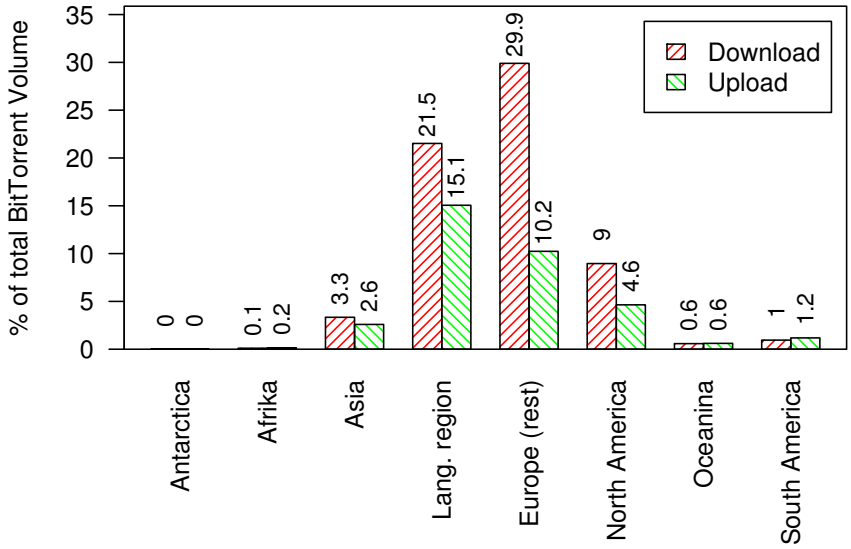


Figure 3.5: Normalized BitTorrent traffic volume.

Table 3.3: Cacheability of NNTP articles.

Cacheability	APR09	AUG09	NNTP-15d
by number of requests	2.0 %	2.6 %	7.0 %
by volume	2.1 %	2.7 %	6.9 %

For eDonkey the cacheability results for peers per file and transferred blocks are not quite as good. For a 24h trace from September 2008 we find that the cacheability per file is 71 % within the AS. However, when considering blocks the cacheability again drops substantially to roughly 4 %.

3.4.2 NNTP

Our caching analysis of NNTP shows that the majority of all articles is requested exactly once, see Table 3.3. We find that less than 7 % of all articles are downloaded multiple times. With respect to volume we observe similar results—the cacheability is again less than 7 %. While we cannot identify a temporal trend cacheability increases with the length of the observation period, i. e., NNTP-15d, and thus also the number of different DSL lines using the NNTP protocol. However, we note that only a small number of lines, less than 2 %, are using NNTP. However, NNTP users are usually among the top volume users.

Table 3.4: Effective cache control headers.

Cache control header	Frequency
`Cache-control`	57.2 %
`Pragma`	0.5 %
`Expires`	1.7 %
`Etag`	21.8 %
`Last-Modified`	6.8 %
none	12.0 %

Table 3.5: Cacheability of top 15 domains (by bytes)

type of service	UGC	fraction	*complete*	*full*	*realistic*
OCH1	✔	12.6 %	2.6 %	0.0 %	1.5 %
OCH2	✔	1.3 %	6.0 %	1.1 %	2.0 %
OCH3	✔	1.0 %	7.1 %	0.0 %	0.2 %
OCH4	✔	0.9 %	2.6 %	0.0 %	0.1 %
Video1	✔	10.8 %	13.0 %	0.0 %	3.8 %
Video2	✔	2.2 %	43.4 %	0.1 %	5.2 %
Video3	✔	1.4 %	7.0 %	0.0 %	0.2 %
Video4	✔	1.4 %	14.7 %	1.5 %	3.6 %
Video5	✔	1.1 %	11.7 %	2.5 %	9.0 %
Software1		2.8 %	63.0 %	68.6 %	64.8 %
Software2		1.8 %	22.0 %	2.9 %	87.4 %
Software3		0.9 %	12.9 %	3.3 %	54.7 %
Software4		0.8 %	56.9 %	42.7 %	65.4 %
CDN1	?	1.5 %	34.8 %	12.8 %	25.4 %
Search	?	1.0 %	56.0 %	5.3 %	32.7 %
overall		100.0 %	21.0 %	9.5 %	21.7 %

3.4.3 HTTP

Given the number of different HTTP scenarios we first discuss some general observations. Next, we explore if cacheability differs by Web service, how it scales with population size, and what might be possible cache optimizations. In the following we report results only for HTTP-14d as the results are consistent across all traces.

General Observations: In principle, there is substantial potential for caching HTTP (see Table 3.6). In the *ideal* scenario 71 % of the requests are cacheable and 28 % of the bytes. This is consistent with previous results [23] which also observe a significantly higher request hit rate than byte hit rate.

Disabling caching across different second level domains, scenario *domain*, does not decrease the caching efficiency by much. It is still 71 % for requests and 27 %

Table 3.6: Overall HTTP Cacheability

	ideal	domain	complete	full	realistic
Bytes	28 %	27 %	21 %	9.5 %	22 %
Requests	71 %	71 %	57 %	16 %	47 %

for bytes. Disabling caching across different URLs for the same object, scenario *complete*, causes the efficiency to drop to 57 % and 21 %. This shows that identical objects are usually not hosted by different providers, while for each provider it is quite common to host the same object on different hosts or with different paths.

Including cache control headers, the scenario *full*, reduces cache efficiency drastically to 16 % and 9.5 %. However, in the *realistic* scenario overall cacheability increases to 47 % and 22 %. The omission of the object ID is responsible for this increase in cacheability: *(i)* the cache may be allowed to serve an object that has changed on the server; *(ii)* aborted downloads and partial requests lead to different object IDs and are thus only cacheable in the *realistic* scenario.

The results of Erman et al. [19] for the *ideal* scenario are even more promising. They found a cacheability of 92 % for requests and 68 % for bytes. Their final results after considering cache-control also show a substantial drop but again indicate a better cache hit rate with 32 % of the bytes. One reason for the more promising results are that they assume that the size of the download is equal to the `Content-Length` header. However, we find that many downloads are interrupted prematurely. In particular large downloads are often aborted. We find that this can lead to measurement error of over 40 % in download volume. Moreover, Erman et al. only consider the `Cache-Control` header. However, almost one third of the replies with cache control do not possess a `Cache-Control` header, but are controlled by `Expires`, `Pragma`, `Etag`, or `Last-Modified`, as listed in Table 3.4.

Individual Sites: Table 3.5 shows the cacheability for the top 15 domains (by bytes) classified according to the Web service that they offer for the scenarios *complete*, *full*, and *realistic*. In column UGC we mark if a domain is dominated by user generated content. When comparing scenarios *complete* and *full* we see that respecting cache control headers often has a devastating effect on cacheability for some domains. When comparing scenarios *full* and *realistic* we see that the negative impact of the object ID also differs across sites. The site Software1 differs: realistic cacheability is lower than predicted by the *full* scenario. This can occur if an intermediate HTTP request invalidates the cache before allowing access to a resource, e. g., when delivering a login page instead of the requested object upon missing authentication cookies.

There are significant differences among the Web hosters. Sites offering software appear to ensure good cacheability (e. g., > 50 %) and can take advantage of caching (e. g., > 60 % for the *realistic* scenario). Some sites hosting videos have substantial potential for caching (> 40 %) but do not take advantage of it. CDNs

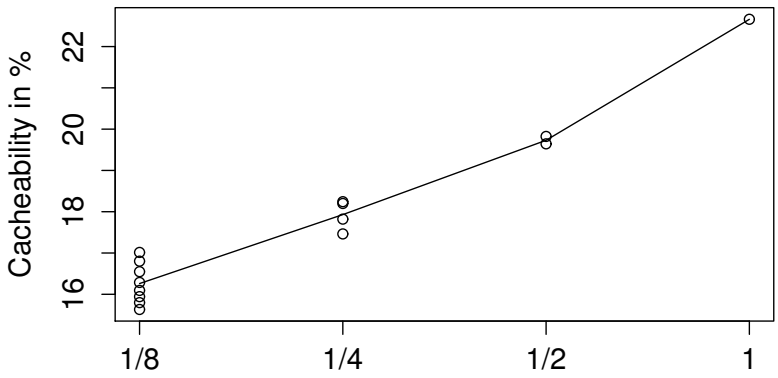


Figure 3.6: Cacheability dependent on fraction of population

also have potential but realize it only partially (34.8 vs. 12.8 %). One click hoster (OCH) have hardly any caching potential (< 8 %) and do not even take advantage of the little potential. The caching hit rate for the *realistic* scenario is less than 2 %. We observe that sites dominated by user generated content exhibit considerably lower cacheability than other sites, e. g., software hosters.

Population Size: Next, we explore the impact of the population size on cacheability. We randomly subdivide our population into smaller sub-populations and recompute the cacheability for the *realistic* scenario. We observe that cacheability appears to increase with population size. When doubling the population size the increase in cacheability ranges from 1.6 % to 2.9 %, cf. Figure 3.6. We presume that the caching potential further increases with an increase in population. However, there may be saturation effects. Also note that the variability of cacheability increases with decreasing population size.

Cache Optimizations: Next we explore why the potential for HTTP caching is not used. For this purpose we allow violations of the strict caching semantics for two high-volume sites. For Video1 we study the impact of personalization and load balancing over servers with different host names. We start at a baseline of 3.8 %. Removing personalization, i. e., parameters, from URLs yields an increased cacheability of 20.1 %. Unifying host names increases cacheability to 24.6 %. Thus, we conclude that personalization can be a major cause for non-cacheability of objects.

Some objects do not include any information regarding their cacheability. Thus in principle they cannot be cached. This may be either intentional, or by negligence of the operator. We now explore if opportunistic caching, meaning setting artificial expiry times and thereby violating strict cache semantics, can help for such objects. More specifically, we examine expiry times of 10 s, 10 min, 1 h, 1 d, and infinite.

The overall effect of opportunistic expiries is only small: 2.6 % increase for a ten minutes timeout and 4.0 % for infinite caching. However, OCH1 and CDN1 show

Table 3.7: Opportunistic caching for the top 15 domains (scenario *realistic*). Domains with no improvement are not shown.

		improvement (percentage points)				
type	baseline	10 s	10 min	1 h	1 d	∞
OCH1	1.5 %	8.1 %	16.6 %	17.0 %	17.8 %	18.4 %
OCH2	2.0 %	0.0 %	0.1 %	0.1 %	0.1 %	0.1 %
Video4	3.6 %	0.2 %	1.0 %	1.2 %	1.3 %	1.3 %
Video5	9.0 %	0.0 %	0.1 %	0.1 %	0.3 %	0.4 %
Software3	54.6 %	0.0 %	0.0 %	0.0 %	0.0 %	0.1 %
CDN1	25.4 %	0.1 %	3.0 %	9.5 %	19.5 %	23.7 %
Search	32.7 %	0.0 %	0.3 %	0.5 %	0.7 %	0.7 %
overall	21.7 %	1.1 %	2.6 %	2.9 %	3.5 %	4.0 %

a large increase in cacheability. For OCH1 even ten minutes is sufficient to gain most of the benefits. Further investigation shows that mis-configured download accelerators are responsible. Such accelerators download large objects across multiple parallel connections. While this is not per se harmful, these accelerators issue partial requests for overlapping regions. As soon as the desired data is fetched, the accelerator closes the connection. However, it takes time to cancel the transaction, and therefore additional data is downloaded. We observe clients that open up to 300 parallel connections resulting in an increase of the download volume by a factor of three. With some cache tuning such extra downloads can be eliminated with a small opportunistic timeout.

3.5 Summary

Our analysis of 20,000 residential broadband DSL lines of a large European ISP shows that contrary to recent work caching is not necessarily beneficial. For NNTP and some Web domain classes, including sites dominated by user generated content, we hardly find any potential. However, some Web service provider can take advantage of caches, e. g., software download providers or CDNs have substantial potential even if unused due to cache control, personalization, and load balancing.

Caching for P2P protocols is in principle very promising, especially when combined with P2P neighbor selection strategies. However, taking advantage of the potential is non-trivial as the cacheability for simple chunk downloads drops to 9 %.

4 DNS in the Wild

In the former chapter we find that caching is promising for P2P protocols, but challanging for the far more prominent HTTP. Many HTTP based content delivery infrastructures operate their servers at various locations, thus offering choice of the download location. Therefore, we next investigate the role of the server assignment algorithm and how it interacts with the user.

4.1 Introduction

The Domain Name System (DNS) was originally intended to provide a naming service, i. e., one-to-one mappings between a domain name and an IP address. Since then, the popular applications have changed from static content hosting to distributed and dynamic content delivery. As a consequence, DNS is a highly scalable system that fulfills the needs of applications that often have very strong requirements in terms of responsiveness of DNS [45, 46, 53]. The scalability of the DNS system stems from the heavy use of caching by DNS resolvers [40].

Today, the DNS system has become a commodity infrastructure that allows applications to map individual users to specific content. This behavior clearly diverges from the original purpose of deploying DNS, and is sometimes considered as abusing it [97]. Given the importance of DNS for end-user experience and how much the DNS system has changed over the last decade, in this chapter we study DNS installation in commercial ISPs and compare it with widely used third-party DNS resolvers, GoogleDNS and OpenDNS.

Based on active measurements carried out across more than 50 commercial ISPs, spanning 5 continents and 28 countries around the world, we study the responsiveness and the returned IP addresses by the local DNS resolvers as well as GoogleDNS and OpenDNS. Our results show that a surprisingly high number of commercial ISPs suffer from poor latency to the local DNS resolver. In general, our results do not reveal drastic differences between the local DNS resolvers, GoogleDNS, and OpenDNS, in terms of responsiveness. Several ISPs show clear signs of DNS load balancing, that leads to a poor usage of DNS caching.

However, our findings reveal that third-party DNS resolvers do not manage to direct the users towards content available within the ISP, contrary to the local DNS ones. We conjecture and partly validate that the reason for this behavior of third-party DNS resolvers has to do with their location, typically outside ISPs, in combination with current inability of DNS resolvers to indicate the original IP

subnet of the end-host in the DNS requests [14]. The current advantage of local DNS resolvers is their ability to represent the end-user in terms of geographic location and its vicinity to content.

In spite of the importance of DNS, we are not aware of any work that has performed such an extensive study of the DNS system based on measurements from end-hosts, and compared local DNS resolvers against third-party DNS resolvers.

The remainder of this chapter is structured as follows: we start with an overview of the DNS system (Section 4.2) and describe our data set and how it was collected in Section 4.3. Then Section 4.4 analyzes our results before we conclude in Section 4.5.

4.2 Domain Name System

A general introduction to the DNS protocol can be found in Section 2.2.1. In this section we concentrate on a brief overview of how DNS is being used today.

When DNS was introduced in 1983, its sole purpose was to resolve host names into IP addresses in a more scalable fashion than the until then used `hosts` file. Since then a number of features and new uses have found their way into the omnipresent DNS. In addition to the increasing complexity within the DNS protocol itself [96], new and oftentimes unforeseen (ab)uses have been established. Paul Vixie heavily criticized those new uses [97]. The most important points are as follows:

CDN load balancing: Content delivery networks set short TTLs on their DNS answers to allow for short reaction times to load shifts, thus crippling cacheability and scalability of the whole system. In addition, CDNs tailor their reply for the IP address of the requesting resolver using the often misguided assumption that the DNS resolver is close to the client originating the request.

NXDOMAIN catching: Some ISPs and also OpenDNS mangle a negative reply with the NXDOMAIN status code into a positive one with the IP address of a search Website under the control of the ISP resp. OpenDNS. By hosting advertisements along the search results it is easily possible to increase the profit margin. While this may work to some degree for Web browsing, applications relying on proper delivery of NXDOMAIN records, e. g., email, are inevitably hampered.

A third-party ecosystem around DNS has evolved over the last couple of years. Players such as OpenDNS, AdvantageDNS, UltraDNS, and most recently Google offer open resolvers to anyone with different feature sets. OpenDNS Basic does NXDOMAIN catching but offers phishing and botnet protection for free. Furthermore, OpenDNS increases the service level for payment between 5 dollars

a month up to several thousand dollars per year for business customers. When Google Public DNS entered the market, their highest-valued goals were to "speed up your browsing experience" and to "improve your security." To achieve both targets Google advertises an impressive list of optimizations and fine tuning [32], e. g., prefetching, load balancing with shared cache, validity checking, and nonce prepending. Google Public DNS also refrains from utilizing NXDOMAIN to make profit. From an implementation perspective, most if not all of the third-party resolvers host their DNS servers on multiple sites around the globe and use anycast to guide DNS clients to the nearest resolver.

In this open market space a user annoyed by his ISP's DNS can easily choose for cost-free third-party service. Tools such as namebench [61] might help him in choosing a well-performing one. The irony, however, is that a user by choosing a different DNS than the one assigned by his ISP will most likely undermine the traffic matrix optimizations performed by CDNs and ISPs, and can potentially even lower his quality of experience due to longer download times.

4.3 Measurements

Data collection on end-hosts is intrinsically difficult due to lack of willingness to participate or due to privacy concerns [38]. Yet, for this chapter we are interested in the DNS performance as perceived by end-users and therefore make efforts to collect data directly from users' computers. This section describes our data and how it was collected.

To achieve our goal of comparing the responsiveness of various DNS resolvers, we wrote a script that performs DNS queries for more than $10,000$ hosts. Amongst other tasks, the script measures DNS response times and returned TTLs for all queried hosts, relying on different DNS resolvers. We asked friends to run our script in early April 2010, leading to traces from more than 60 vantage points, covering 28 countries and 5 continents. Overall, we have obtained traces for some 50 commercial ISPs. During our measurements, the following information was collected:

1. **Vantage point:** Our script initially determines the public IP address and operating system of the executing machine as well as a current time stamp and the IP address of the local DNS resolver.

2. **Resolver:** Periodically, we determine the RTT[1] towards the local, Google, and OpenDNS resolver and perform traceroutes towards these three resolvers. This reveals potential route changes and the proximity of DNS resolvers to our vantage points.

[1] We rely on the response time reported by `dig` when querying for the root zone NS records, rather than using `ping` or `traceroute`.

3. **Host:** For each of our approximately 10,000 target host names we perform, using the `dig` program, two consecutive DNS queries and measure the response times. Comparing response times between the first and second query gives insights into caching and load balancing, see Section 4.4.2. Besides response times, we record the returned TTL values and the IP addresses of the DNS responses.

Presumably, the majority of Internet users rely on DNS services that are provided by their ISP. This local DNS resolver is automatically configured during the dial-in handshake or via DHCP (*Local DNS*). Yet, alternative DNS providers claim to speed up the browsing experience (GoogleDNS) [32], and some users think that DNS queries can be processed much more quickly by employing a large cache of domain names (OpenDNS) [63]. To check for potential differences in performance, our script sends the same DNS queries to multiple DNS servers: the locally configured DNS server, to 8.8.8.8, and to 208.67.222.222, whereby the latter two are the DNS IP addresses used by Google and OpenDNS[2], respectively.

In order to improve its efficiency, DNS heavily relies on caching, i. e., storing DNS query results for a period of time in DNS cache servers provided by ISPs or in home routers that implement DNS caches. As we seek to investigate potential bias in DNS response times for different types of queried hosts (e. g., popular vs. rarely queried hosts), we download from Alexa the list of top 1,000,000 sites [2], and select more than 10,000 hosts to be queried by our script as follows:

top5000: These are the 5,000 most popular hosts according to the Alexa ranking. The answer to many of these DNS queries may already be stored at close-by cache servers. We point out that the top5000 hosts are selected based on a global ranking, and hence are not necessarily the most popular hosts if ranked by country, region, etc.

tail2000: These are 2,000 hosts from the tail of the Alexa ranking and are less likely to be cached in close-by DNS servers.

embedded: Many Web-pages include embedded content (e. g., AVI, Flash, or Java) that the browser may have to retrieve separately from different domains. We take the top 1,000 hosts according to the Alexa ranking, download with `wget` the content of all these hosts and compile a list of domains from which such embedded content is retrieved. By doing so, we obtain some 3,500 host domains.

Restricting to 10,000 hosts allows our measurements to finish within a couple of hours, which turned out to be acceptable to our end-users. Resolving the names of our 10,000 hosts reveals that a considerable fraction of them (709) rely

[2]Other DNS IP addresses such as 8.8.4.4 for GoogleDNS are generally configured as secondary DNS server.

on DNS redirection. The set **redirected** contains all host names for which we see a CNAME record to an external domain (such as a CDN). The set **akamaized** is a subset of **redirected**, containing the 434 hosts that are redirected to Akamai. Information about redirection will be used for the CDN study in Section 4.4.3.

In principle, there can be interactions between two different vantage points in our experiments if the script is run close in time and based on the same list of hosts: for example, OpenDNS or GoogleDNS can cache the answer when vantage point *A* sends a query. When another vantage point *B* sends the same query, the response time will be significantly shorter if the reply is already in the cache. However, by inspecting timestamps in our traces and the DNS servers' approximate locations as revealed by traceroute and the RTT, we can infer whether interactions may have happened. The traces presented here are carefully selected and do not show any degree of interaction.

4.4 Evaluation of DNS resolvers

In this section we rely on the measurements explained in the previous section, and analyze the behavior of the different DNS resolvers, with respect to responsiveness (Section 4.4.1), DNS installation (Section 4.4.2) and the returned answers (Section 4.4.3).

4.4.1 Responsiveness

Google claims on its Website that Google Public DNS speeds up browsing performance [32]. One primary goal of this chapter is to understand the impact of the selected DNS resolver on the observed DNS response time. Is it really true that alternative DNS resolvers such as GoogleDNS or OpenDNS offer better performance than the local resolver?

To answer this question, it is crucial to rely on measurements that are carried out directly from end-hosts connected to commercial ISPs, see Section 4.3. If the DNS installation within the local ISP is properly done, we would expect very small latencies to the resolvers maintained by the local DNS. Yet, we find cases where GoogleDNS and OpenDNS outperform the local DNS resolver in terms of observed response times.

We select two vantage points that are representative for many other traces. The first is located in Germany and we qualify it as "good ISP". The second is based in the US and we call it "bad ISP". It is not our goal to assess individual ISPs. Rather these terms reflect that the "good ISP" shows better DNS performance in terms of response times, load balancing, caching, etc., compared to the "bad ISP", see Section 4.4.2. Figure 4.1 and Figure 4.2 display the CCDF of the response times, in milliseconds, observed at these two vantage points. The leftmost part of the curves on these two figures shows the minimal latency that has been achieved by

the DNS resolvers across all 10,000 queries. This minimal latency can be seen as a metric to characterize the proximity of a DNS resolver to the actual end-host. Figure 4.1 shows a case where the smallest latency differs significantly between the DNS resolver of the local ISP, OpenDNS, and GoogleDNS at 11ms, 24ms and 44ms. Although both GoogleDNS and OpenDNS maintain a large set of strategically placed resolvers and rely on anycast to route DNS queries, their latencies are far higher than those of the local resolver. The local resolver appears to be close to the end-host. This underlines the importance of placing a resolver in the proximity of end-hosts.

Surprisingly, there are cases where we observe that GoogleDNS or OpenDNS perform as well if not better than the local ISP resolver, see Figure 4.2. For our "bad ISP" the network distance towards OpenDNS appears to be especially small. Indeed, the RTT towards OpenDNS is only 10ms while it is 11ms towards the local DNS. In total, we observe 17 vantage points where either GoogleDNS or OpenDNS have in the worst case the same latency as the local DNS. On 21 vantage points, the local DNS is at least 25ms faster than the other two third-party resolvers; for the remaining 29 vantage points the local DNS only marginally outperforms GoogleDNS and OpenDNS.

In addition to deploying resolvers in the proximity of end-hosts, another key aspect to achieve good DNS performance is efficient caching. With respect to caching, Google aims to increase the number of cache hits through load balancing DNS resolvers that avoid cache fragmentation or by actively prefetching names [32]. When the curves for the first query are close to vertical, this shows that the caches are primed. Based on our plots, the three resolvers do not seem to be so well primed. While GoogleDNS performs considerably better on the tail of the curve than OpenDNS for the traces shown, this does not hold in general. Based on our measurements, we can neither confirm nor refute any gains obtained from techniques such as name prefetching or load balancing for shared caching as Google or OpenDNS may use.

To study caching behavior, our measurements always perform two consecutive DNS queries for the same name. Comparing the curves in Figure 4.1 for the first and second DNS query, we observe considerably faster response times for the second query due to caching of the DNS answers by the resolvers. The differences in the latencies to the resolvers become then even more obvious. Typically over 95% of the second queries are being answered within 100ms.

Overall, the barrier to achieve lower DNS response times seems to be the distance to the local resolver, once the DNS resolver cache is properly populated. GoogleDNS for instance does not seem to achieve their 50ms objective [46] for most of the queries in the case of the "good ISP" (Figure 4.1). There are still less than 1% of the cases for which all resolvers take more than 100ms to answer, due to non-cacheable records, one-time errors, and measurement artifacts[3].

[3]Typically, the second latency is considerably higher than the first one for at most 10 out of 10,000

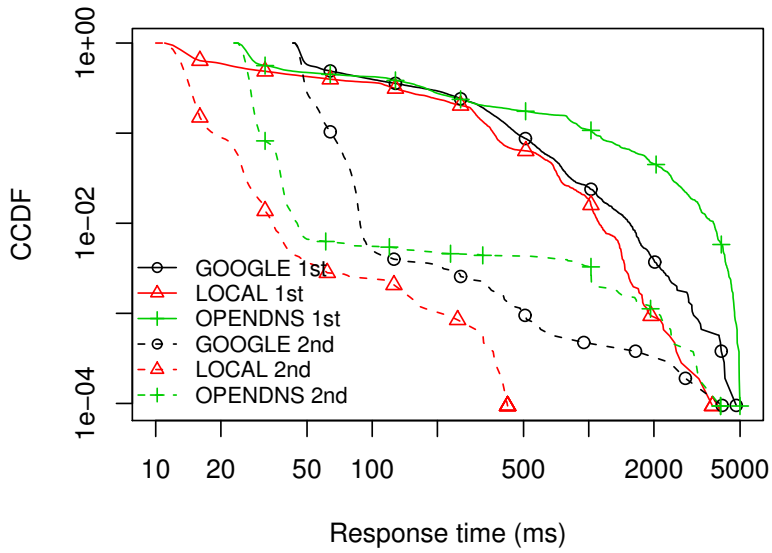


Figure 4.1: CCDF of response times for "good ISP". The local resolver has significantly lower RTTs than both GoogleDNS and OpenDNS.

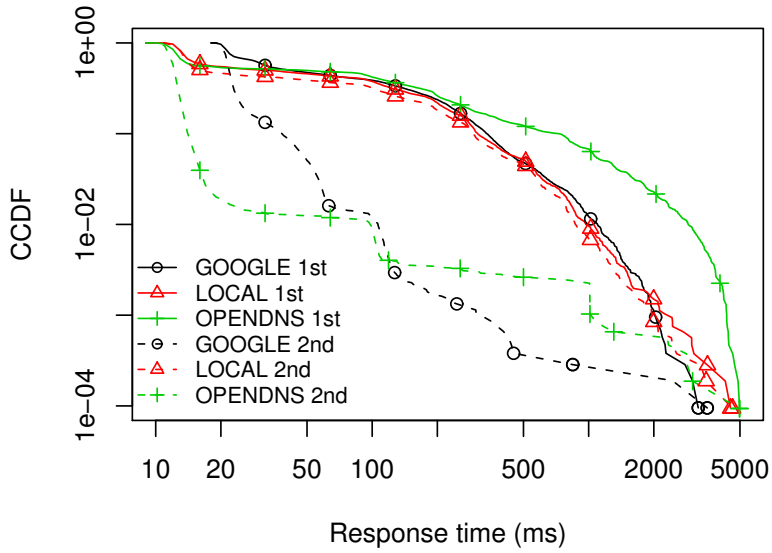


Figure 4.2: CCDF of response times for "bad ISP". The time for the second query does not improve much.

4.4.2 Resolver Infrastructure

The observation from the previous section, that most of the second queries can be answered from the cache, does not hold for the local DNS resolver of the "bad

hosts. Potential reasons may include fluctuations in routing, links resets, server reconfiguration, etc.

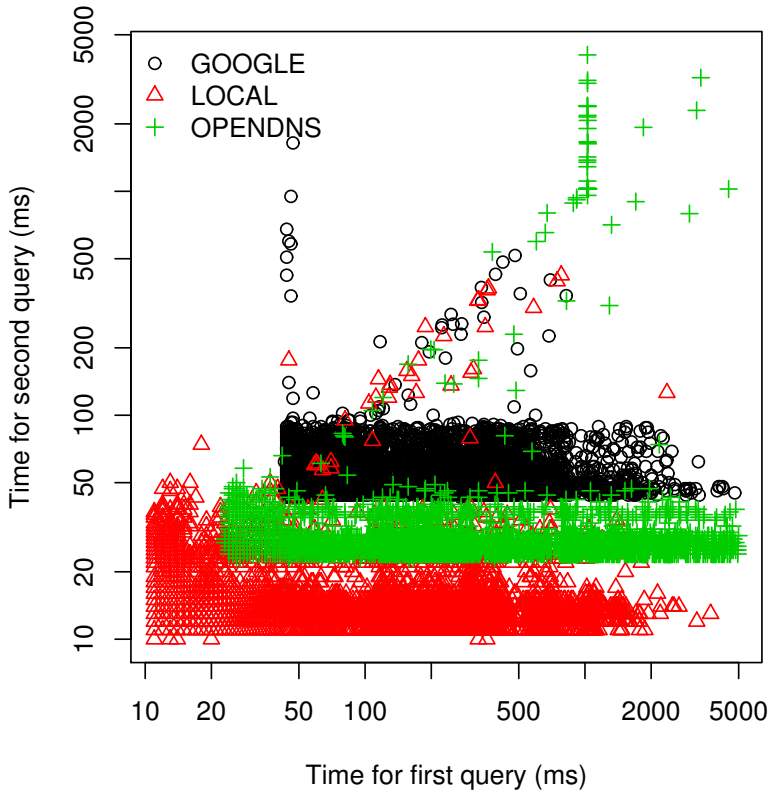


Figure 4.3: "Good ISP": as expected most of the second queries can be answered significantly faster than the first query due to caching.

ISP" (Figure 4.2). In this section, we dig further into the results from Section 4.4.1, by showing the results from the first query against those of the second query on a scatter plot (see Figure 4.3 and 4.4). The x-axis of Figure 4.3 and 4.4 show the response time in milliseconds for the first query, while the y-axis the response time in milliseconds for the second query.

In Figure 4.3, we observe one horizontal line per DNS resolver for the "good ISP", meaning that the response times for the second queries show only small variation and are consistently better than those for the first query. An ISP that has a properly deployed DNS infrastructure should show this kind of pattern. However, several of our vantage points display a behavior like the "bad ISP" (Figure 4.4), where points are scattered along a horizontal and vertical line, as well as the diagonal. We explain this behavior by a load balancing setup without a shared cache. Sections with a sharp decline in the CCDF for the second query (Figure 4.1) correspond to the horizontal patterns in Figure 4.3: the first query primed the cache and the second query could be served from that cache. The diagonal in Figure 4.4 stems from hostnames for which both queries needed an iterative resolving because the second query was redirected to a different re-

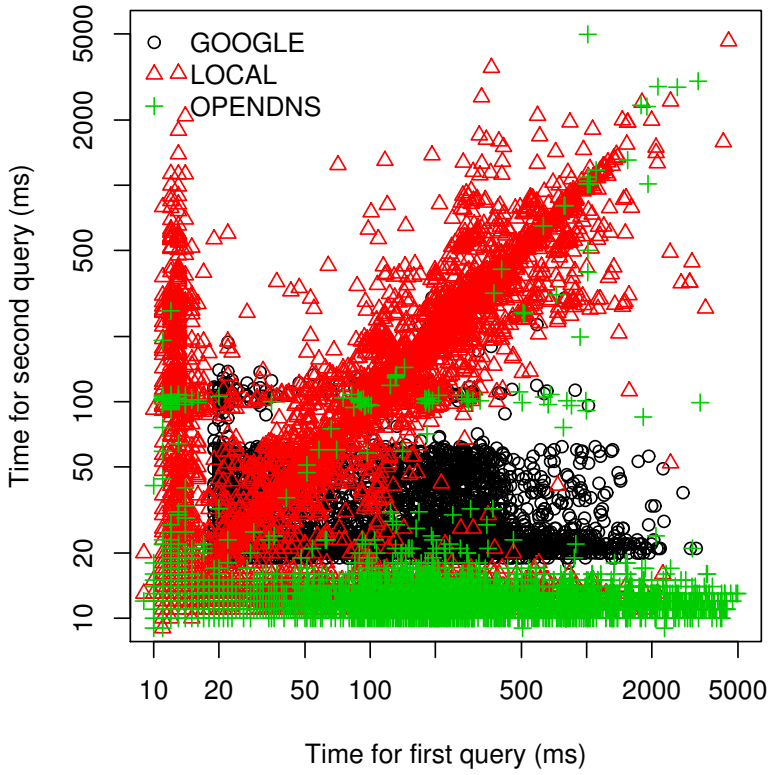


Figure 4.4: "Bad ISP": the ISP balances load over different DNS resolvers, so the second query cannot always be answered from the cache. A strong diagonal and a vertical line emerge.

solver. Finally, the vertical line springs from host names for which the first query could be served from the cache, while the second query was directed to a different resolver where the cache was not primed. Our conjecture is supported by consistent observations in which a significant proportion of the TTLs for the first and the second query differ considerably.

Several ISPs for which we have multiple traces display this behavior in a consistent way. Furthermore, we see this behavior for both OpenDNS and GoogleDNS in several traces. We conjecture that OpenDNS and GoogleDNS also use load balancing for highly loaded sites.

For some ISPs, we observe high RTTs towards the local DNS *and* load balancing. We conjecture that in these cases the DNS infrastructure is centralized and requires load balancing to compensate for the high number of queries arriving at a single location.

Although there are valid reasons to rely on DNS load balancing, the way some ISPs are implementing it prevents caching from being properly utilized. A hierarchical DNS infrastructure could improve hit rates while preserving distribution

of load to different machines.

4.4.3 Comparing DNS Answers

For good end-user experience, fast DNS response times are important. While this aspect has already been studied in Section 4.4.1, we now investigate the exact answers that DNS returns, i. e., the resolved IP addresses. Frequently, there is more than one available option from where content can be retrieved. Possible reasons include both the use of load balancing and content distribution networks (CDNs) [45, 93]. The latter replicate content and provide copies near the client to optimize network performance [53]. The goal of this section is to study the observed diversity in resolved IP addresses for each DNS name across different vantage points and different DNS resolvers. In particular, we examine potential interferences between the choice of the DNS resolver and measurements done by CDNs. After all, CDNs such as Akamai determine which IP to return in the DNS response to the client based on measurements done against the DNS resolver, not against the client. Choosing a DNS resolver that is far from the end-host might vitiate the performance optimizations made by CDNs.

As expected, there is indeed diversity in the IP addresses returned by DNS. While we perform DNS queries for $10,000$ unique host names in our experiments, the overall number of unique resolved IP addresses across all traces is $36,000$. One reason is that we always perform two DNS queries for the same host name and then even repeat this for three different DNS resolvers. Apart from load balancing, this can be due to CDN content. When repeating queries or when changing between DNS resolvers, the resolved DNS names may be different depending on the mechanisms CDNs use to optimize network performance for the clients.

In Section 4.4.1 we find that the local DNS resolvers generally provide lower latencies due to their proximity to the end-hosts. Hence, one may speculate that they better represent the location of end-hosts than other resolvers. If possible, it might be more desirable for performance or economic reasons that CDNs be queried by a DNS resolver that is located within the local ISP of the end-host. Figure 4.5 shows for each vantage point of our study (x-axis), how many IP addresses that belong to the same ISP as the host of the vantage point, were returned by each DNS resolver, across all queried content.

We find that the majority of DNS answers point to content outside the vantage point's network. GoogleDNS and OpenDNS even return IP addresses from different networks for all our traces. One of the reasons is that these resolvers are usually not located inside the ISP. Yet, for approximately 30 vantage points we observe that content is downloaded from at least 100 hosts located within the ISP's network when the local DNS resolver is queried. There even exist vantage points where local access occurs for 926 of our $10,000$ host names.

Although this may not appear much, it is significant if we consider that this

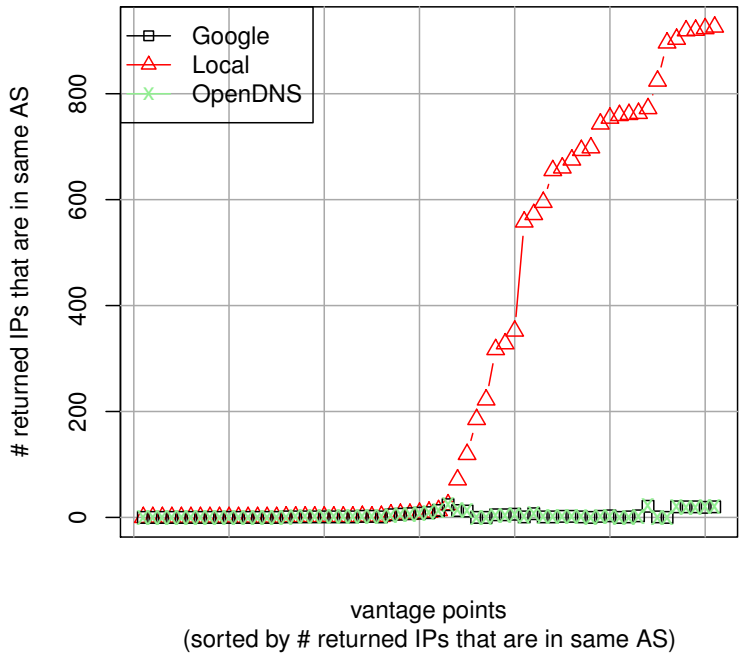


Figure 4.5: Number of DNS answers directing a client to a local replica of content.

locally available content completely covers the `akamaized` set[4] (see Section 4.3). We harvest IP addresses of Akamai servers by sending DNS queries to Akamai content from different Planetlab servers. Interestingly, we find based on manual inspection that the vantage points with local content generally have an Akamai server deployed within the same network.

From Figure 4.5 we can infer that only local DNS resolvers direct end-users to content that is locally available in the network of the vantage point. Instead of matching the results of the resolvers against the network of the vantage point, we now directly compare our three DNS resolvers pairwise against each other. More precisely, we count how often the results of the two resolvers are different with respect to the subnet, autonomous system (AS), and country[5] of the DNS answer. Figure 4.6 presents for each vantage point of our study (x-axis) the number of differences for the comparison local DNS vs. GoogleDNS.

Figure 4.6 reveals that the answers to the DNS resolvers differ in terms of subnets for approximately $2,000$ out of our $10,000$ host names. In half of these cases, the returned IP addresses even belong to different ASs and countries. Since the local DNS resolver points to content inside the ISP's network for a significant number of host names (Figure 4.5), we claim that GoogleDNS and OpenDNS un-

[4]Additionally, there is strong overlap with `top5000` and `embedded`.
[5]We used geolocation data to map IP addresses to countries [36].

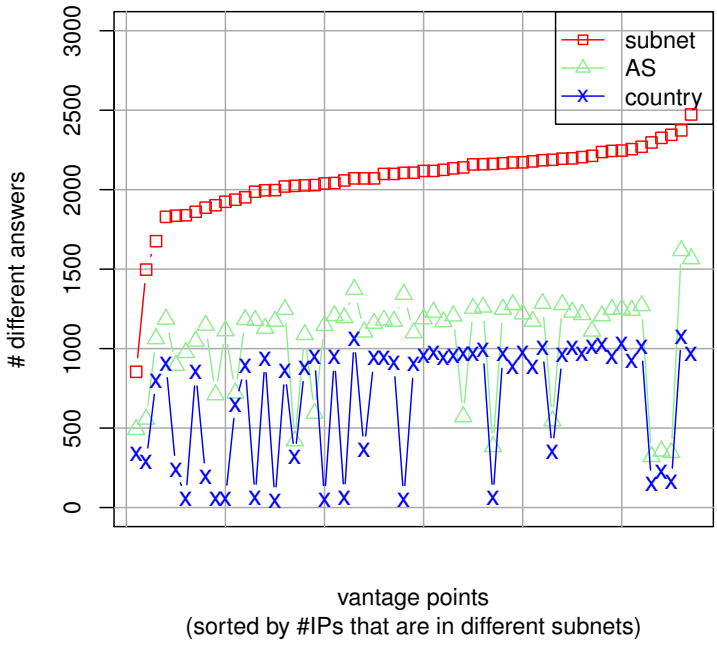


Figure 4.6: Number of differing DNS answers when comparing the local resolver with GoogleDNS.

necessarily direct end-users to content servers in different ASs or even subnets. We do not present the plots for the comparisons between Local vs. OpenDNS and Google vs. OpenDNS. Both plots are very similar to Figure 4.5. Apparently, whenever we change the DNS resolver, there will be different answers from DNS for at least some of the host names. This observation justifies recent activities of the IETF in the direction of standardizing a way to include the subnet of the original end-host in DNS requests [14].

4.5 Summary

Based on active measurements from inside more than 50 commercial ISPs, we have studied DNS performance by comparing the ISPs' DNS installation against widely used third-party DNS resolvers, namely GoogleDNS and OpenDNS.

Typically, end-hosts experience very small latencies to the resolvers maintained by the local ISP, though there exist cases where GoogleDNS and OpenDNS outperform the local DNS resolvers in terms of the observed response times. Moreover, our findings suggest that several ISPs and OpenDNS rely on a load balancing setup without a shared cache, resulting in poor caching efficiency. Even Google Public DNS, despite their claim [32] exhibits the same behavior for a few

vantage points. Moreover, we observe that third-party DNS resolvers do not manage to direct the users towards content available within the ISP, contrary to the local DNS ones. This observation holds for all akamaized content.

5 Who is Who in Content-land?

So far we have looked at the detailed protocol level (Chapter 3) and at the impact of the DNS resolver on the server selection process (Chapter 4). We found that caching, per-se, is still hard, and that the selection of the content cache is sensitive to the choice of DNS resolver. In this chapter, we complement our findings with a high-level viewpoint. We ask the question: From where is content in principle available?

5.1 Introduction

Today's demand for Web content in the Internet is enormous. The sheer volume of data that has to be delivered to end-users has triggered the emergence of a diverse set of hosting and content delivery infrastructures. Recent traffic studies [19, 49, 56] show that Web-based applications are again very popular. To cope with the increasing demand for content, Web-based applications and Web content producers use scalable and cost-effective infrastructures. These infrastructures, which we refer to as *hosting infrastructures* throughout this chapter, have multiple choices on how and where to place their servers. Some choose to rely on a single location, others on planetary-scale fine-grained installation, or any other option in-between.

The recent work from Labovitz et al. [49] observed, by monitoring the traffic of a large number of ISPs, a trend in application and content consolidation. They report a shift of traffic away from what used to be the core of the AS-level topology, i. e., tier-1 carriers, in favor of content-centric organizations such as Google.

In this chapter, we propose *Web content cartography* and take the first few steps in this direction. Web content cartography consists of building an abstract representation of the Web hosting infrastructure at multiple levels, e. g., ASs, prefixes, geographic locations.

As demand drives hosting infrastructures to make a given content available at multiple locations, identifying a particular hosting infrastructure requires sampling its location diversity [90]. We approach Web content cartography through a methodology that samples this location diversity as exposed by hosting infrastructures. Distributed hosting infrastructures rely on DNS redirection to select servers from which end-users obtain content [53]. As the server selection may depend on the location of the DNS resolver of the end-user [57], cf. Chapter 4, end-user based measurements from diverse locations are required to sample the

hosting infrastructure. In this chapter we rely on a large set of vantage points located inside 78 ASs across the Internet. Each vantage point performs active measurements in the form of DNS queries for a large list of popular domains.

Our contributions can be summarized as follows:

- *Maps of hosting infrastructures:* We build maps of hosting infrastructures at multiple levels, e. g., ASs, prefixes, geographic locations.
- *Content replication:* We highlight the prevalence of content replication in the Internet and its impact on local content availability in different regions of the world.
- *Classification of hosting infrastructures:* We identify individual hosting infrastructures and their different deployment strategies.
- *Content potential:* We quantify the relative amount of content that is hosted at a given location, e. g., in an AS or a country.
- *Content Monopoly Index*: We introduce an index that reflects the type of content an organization hosts, either replicated or exclusively hosted.

The remainder of this chapter is structured as follows. We review the state of content delivery in today's Internet in Section 5.2. We present our measurement approach in Section 5.3 and validate our approach in Section 5.4. We study geographic properties of hosting infrastructures in the Internet in Section 5.5. We identify and characterize individual hosting infrastructures in Section 5.6. We map hosting infrastructures to the AS-level, rank the corresponding ASs, and compare our AS rankings with existing ones in Section 5.7. We discuss the implications of our work in Section 5.8. We present related work in Section 5.9 and summarize the chapter in Section 5.10.

5.2 Content: the King of the Internet

In this section, we give a short overview of two particular observations that motivate Web content cartography. First, the recent work by Labovitz et al [49] observes significant shifts in interdomain traffic and peerings. Second, content hosting and delivery has changed significantly during the last few years, especially through the widespread use of DNS by hosting infrastructures to perform server selection from which end-user obtain the requested content.

5.2.1 Internet Traffic and Topology Changes

Today's Internet traffic [49] differs significantly from the one observed a decade ago [12, 20, 92]. The early commercial Internet had a strongly hierarchical struc-

ture, with large transit Internet Service Providers (ISPs) providing global connectivity to a multitude of national and regional ISPs [91].

During the early times of the commercial Internet (mid-90's), most of the content was delivered by client-server applications that were largely centralized. At the time, content was coming mainly in the form of Websites hosted in enterprise data-centers. The tremendous growth of the World Wide Web, video streaming, and user-generated content has transformed hosting and content distribution into commodity services. A large fraction of today's content is hosted and delivered from data-centers and content distribution networks [49, 53].

With the commodification of content hosting and delivery, the Internet traffic and topology landscape has been fundamentally reshaped [49]. Today, a limited number of content contributors are responsible for a large fraction of interdomain traffic. Labovitz et al. [49] observed consolidations both at the AS-level as well as the application-level. These consolidations of the content takes place with changes in interconnection policies between ASs. Nowadays, large content contributors often have direct peerings with large ISPs or are even co-located within ISPs to deliver their content to the users. Therefore, the Internet structure is not as strongly hierarchical as it used to be. Other independent studies confirm these results at the AS-level [16, 31].

5.2.2 Content Delivery in the Internet

Driven by the tremendous demand from end-users for content, a diversity of hosting and content delivery infrastructures has emerged during the last years. These infrastructures, which we refer to as hosting infrastructures, have multiple choices on how and where to place their servers. As described by Leighton [53], the main approaches are *(i)* centralized hosting, *(ii)* data-center-based content distribution network (CDN), *(iii)* cache-based CDNs, and *(iv)* peer-to-peer (P2P) networks. Approaches *(ii)* and *(iii)* allow to scale content delivery by distributing the content onto a dedicated hosting infrastructure. This hosting infrastructure can be composed of a few large data-centers, a large number of caches, or any combination. In many cases, DNS is used by the hosting infrastructure to select the server from which a user will obtain content [45, 90, 97], cf. Chapter 4. In this section, we take advantage of the unique view that DNS offers to sample the different locations of each hosting infrastructure.

Approach *(iv)* , P2P, can be seen as a fully distributed way to deliver content. Despite the widespread popularity of P2P traffic in the early 2000, recent measurements have observed a decline of P2P traffic [19, 49, 56]. Still, in specific parts of the world P2P is still popular and infrastructures to distribute P2P content have been identified [15]. In this chapter we focus only on Web content.

5.3 Approach

In this section we describe our approach for identifying the infrastructures used for hosting and delivering Web content in the Internet. The key idea is to examine, for a large number of geographically distributed vantage points, the IP addresses that DNS returns for various popular names (Section 5.3.1). Section 5.3.2 explains how we compile a list of different types of hosts to be queried at each of our vantage points. Then, Section 5.3.3 explains which information we collectand summarizes how we process and clean our data. Note, our approach shares similarities with the one in Chapter 4.

5.3.1 From Vantage Point Diversity to Server Diversity

For our study, it is crucial to utilize vantage points that reside in different networks and countries. The reason for this relates to the use of DNS by the hosting infrastructure to select the server from which a user obtains the requested content [45, 90, 97], cf. Chapter 4. In particular, CDNs rely on the IP address of the local DNS resolver to determine the IP address that is returned by DNS [57]. The hosting infrastructure optimizes the server selection based on the location of the DNS resolver of the client, assuming that the client is close to its DNS resolver. Therefore, to sample the locations from which a given hosting infrastructure serves content, we need vantage points using different DNS resolvers in different networks, ASs, and countries around the world.

5.3.2 Compilation of Hostname List

Due to the sheer size of the Internet – an estimated 92 million of active domains only for the ***com.*** top-level domain [17] – it is simply impossible to query all host names in the Internet. However, there is high variation in the popularity of Web content. Consolidation of content into data centers or CDNs [49] is likely to even reinforce the trend towards a small number of extremely popular hosts. We use a selection of hostnames which we expect will allow us to sample the hosting infrastructure that serves a major fraction of today's Web traffic. Given that Internet traffic at various levels of aggregation is consistent with Zipf's law [13,20,98,102], the hosting infrastructure that serves popular hostnames is likely to be responsible for a major part of today's Internet traffic.

To obtain a good coverage of the larger data centers and hosting infrastructures, we decide to include in our list hostnames that are top ranked by Alexa [2]. Alexa relies on statistical sampling and determines its ranking by counting how many pages were visited by Internet users who have downloaded their toolbar. Note, Alexa itself is already accounting for various sampling biases of its user list[1]. In

[1] See `http://www.alexa.com/help/traffic_learn_more`.

order to check for potential differences and to scrutinize replication of content also for less popular hosts, we further add hosts that are at the bottom of Alexa's ranking.

Moreover, many Webpages contain embedded content, e. g., images, videos, and advertisements that the browser of the user has to download from different servers. In our study, such embedded content has to be taken into account, as it might be served from servers other than those serving the front page of a popular hostname. To give an example, the front page of `facebook.com` might be served from the Facebook datacenter, but the logo and other embedded objected such as the profile photo might be served from the Akamai content distribution network. To obtain EMBEDDED, we extract the hostnames from the main page of all hosts in TOP2000.

Finally, the global ranking from Alexa [2] does not necessarily contain the most popular hostnames if ranked by country. Therefore, we also obtain from the Alexa Website the regional rankings for five countries (Australia, China, US, France, Germany) and add the 100 most popular hostnames from each country to our list.

Overall, we keep the 2,000 most popular and 2,000 from the least popular hostnames according to the Alexa ranking. In addition, we include more than 3,000 hostnames used for hosting embedded objects, and some regional popular hosts. This list leads to three subsets which we will refer to as TOP2000, TAIL2000, and EMBEDDED respectively for the remainder of the chapter.

5.3.3 Measurements

To collect the measurements, we wrote a script and published it alongside with explanations regarding the project on a Website[2]. We initially announced the project on IMC 2010. In addition we asked on several DNS related mailing lists and our personal friends for support in the measurement campaign. This resulted in a total of 484 raw traces.

For every trace we collect the following information: For each item on our host list we perform a DNS query towards each the local DNS server, OpenDNS [64] and Google Public DNS [32], and store the full replies. We determine the IP address of the vantage point as seen by the outside world after every 100 DNS queries. This enables us to identify users who roam across different networks.

To remove artifacts, we perform a thorough cleanup process on the 484 raw traces we collected. We check for the following measurement artifacts:

- We remove incomplete traces.
- We do not consider traces if the vantage point roams across ASs during the experiment.

[2] `http://www.fg-inet.de`

- When a vantage point provides us with repeated measurements over time, we only use the first trace that does not suffer from any other artifact to avoid bias in our similarity concept, see Section 5.4.3.
- If the DNS resolver of the vantage point is a well-known third-party resolver, e. g., OpenDNS, GoogleDNS, we do not consider the trace. In Chapter 4 we showed that using third-party resolvers introduces bias by not representing the location of the end-user.
- If the DNS resolver of the vantage point returns an excessive number of DNS errors, or is unreachable, we do not consider the trace.

After removing all traces with the above artefacts, we have 133 clean traces that form the basis of this study. For the different aspects we examine we will use either the responses of the local DNS resolvers only (*LocalDNS*), or the responses of all queried DNS resolvers: the local resolvers, OpenDNS, and Google Public DNS (*AllDNS*).

5.3.4 Mapping IP Addresses

Web content cartography requires knowledge about the geographic location of IP hosts and about the AS in which a host is located. For the former, we rely on the Maxmind geolocation database [58]. To determine the AS for a given IP address, we use BGP routing information from RIPE RIS [77] and RouteViews [78], and assume that the last AS hop in an AS path reflects the origin AS of the prefix. Note, throughout the chapter, we rely on both the granularity of BGP prefixes as well as /24 subnetworks. /24 subnetworks have the advantage over BGP prefixes of better representing the actual usage of the address space by hosting infrastructures.

5.4 Validation of Approach

We now validate our choice of hostnames (Section 5.3.2) as well as the coverage provided by our vantage points (Section 5.3.1). In this section it is our goal to gain insight about how to choose vantage points and host lists to achieve good coverage with low overhead. In Chapter 4 we show that a single vantage point will get different replies from third-party resolvers than from his local resolver. Therefore we choose *AllDNS* to maximize the number of IP addresses discovered by each vantage point.

5.4.1 Network and Geographic Footprint of Vantage Points

We map the IP addresses of vantage points of the 133 clean traces to ASs and countries using the mapping methodology described in Section 5.3.4. This leads

Figure 5.1: World map of countries that host vantage points in our experiments.

Table 5.1: Coverage of traces.

# traces	133
# covered ASs	78
# distinct countries	27

to a coverage of 78 ASs and 27 countries (see Table 5.1) that span six continents (see Figure 5.1). Our experiments include traces from major residential ISPs, e. g., AT&T Internet Services, Comcast, Verizon, Road Runner, Telefonica, Deutsche Telekom, British Telecom as well as smaller residential ISPs and some university and research networks.

5.4.2 Network Coverage by Hostname

Our aim is to identify and map Web hosting infrastructures in the Internet. Previous studies [8,25,37,90] were able to achieve an exhaustive coverage of a limited number of well known hosting infrastructures. In our study, we strive to achieve a wide coverage of the prevalent hosting infrastructures without targeting a priori known hosting infrastructures. Our approach has the advantage of being able to identify emerging hosting infrastructures based on the popularity of the content they host.

We next investigate the scope of the network coverage of our study. For this, we analyze to which degree replies for different parts of our hostname list achieve network coverage that is utilized by hosting infrastructures. To identify the IP ranges utilized by hosting infrastructures we aggregate the returned IP addresses

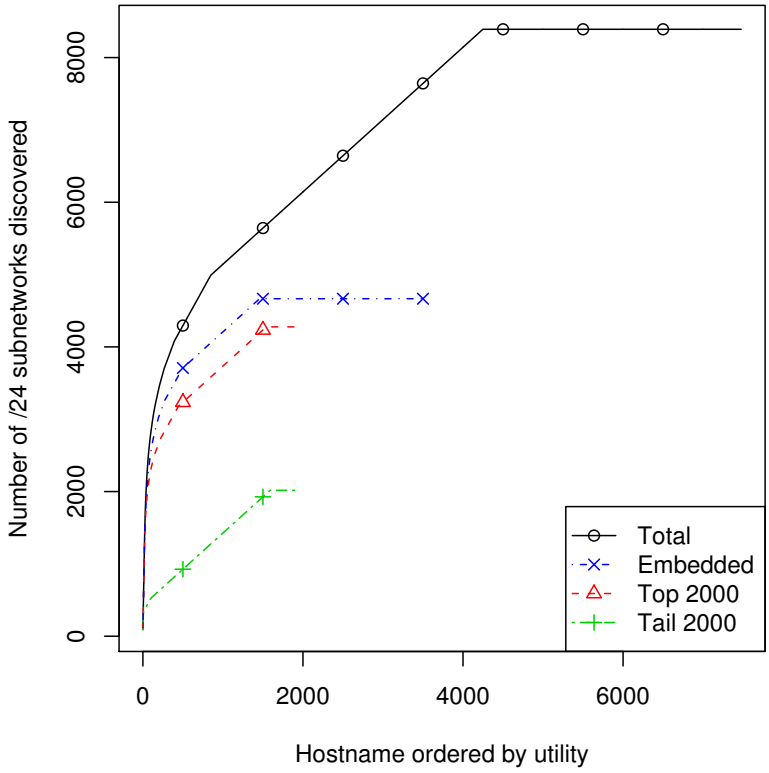


Figure 5.2: /24 subnetwork coverage by hostnames.

over /24 subnetworks. Figure 5.2 shows the additional number of distinct /24 subnetworks of hosting infrastructures when adding hostnames from our list (see Section 5.3.2). The x-axis of Figure 5.2 orders hostnames by decreasing *utility*. We define the utility of a hostname as the number of /24 subnetworks it adds to the set of /24 subnetworks discovered by a set of hostnames. The y-axis of Figure 5.2 shows the total number of /24 subnetworks found as a function of the number of hostnames. In addition to the total, we differentiate between the three types of hostnames introduced in Section 5.3.2: TOP2000, TAIL2000, and EMBEDDED.

The curves in Figure 5.2 can be separated into three regions: a steep slope on the left, followed by a region with a slope of 1, and a flat region at the end. The steep slope region identifies hostnames with a high utility. These hostnames should be included to discover a significant fraction of the content infrastructure with limited probing effort. The region having a slope of 1 results from hostnames that positively contribute to the coverage but whose utility is much lower than hostnames on the left. The third and flat region corresponds to hostnames that return redundant information about the content infrastructure, compared to the first two regions.

Let us now turn to the pairwise comparison of the three types of hostnames.

While of equal size, the /24 subnetworks observed by TOP2000 and TAIL2000 exhibit a difference by a factor of more than two in the number of subnetworks they cover. The same amount of popular content is served from a network-wise much more diverse hosting infrastructure than less popular content. Most of the difference in the cumulative utility between TOP2000 and TAIL2000 stems from a small number of popular hostnames.

We also investigate the overlap between the hosting infrastructure that serves both TOP2000 with the one that serves embedded objects. We see that both are served by hosting infrastructures that have considerable network overlap. Moreover, closer investigation shows that hosting infrastructures that serve both TOP2000 and EMBEDDED are, in large parts, network independent from the one that serves TAIL2000.

5.4.3 Network Coverage by Trace

Hosting infrastructures rely on geographic hints to serve content from servers close to the end-user [53]. We expect that traces in diverse regions of the world sample different parts of the hosting infrastructures. Therefore, we address now the utility of traces obtained from different vantage points.

Figure 5.3 shows the additional number of distinct /24 subnetworks of hosting infrastructures when adding traces from our dataset (see Section 5.3.3). The x-axis of Figure 5.2 orders traces by decreasing *utility*. We define utility of a trace as the number of /24 subnetworks it adds to the set of /24 subnetworks discovered by a set of traces. The y-axis of Figure 5.2 shows the cumulative number of /24 subnetworks found as a function of the number of traces. We also differentiate between the three types of hostnames introduced in Section 5.3.2: TOP2000, TAIL2000, and EMBEDDED.

In total, we find more than 9000 /24 subnetworks that are utilized by hosting infrastructures. We observe that every trace samples about half of these subnetworks (4800). About 2800 of these subnetworks are found in all traces. This relatively high fraction of common subnetworks among traces is the consequence of our choice of hostnames that is not biased towards any given hosting infrastructure.

From a careful investigation, we noticed that the traces that provide the highest utility (traces corresponding to the leftmost side in Figure 5.3), are actually located in different ASs and countries. For example, the first 30 traces belong to 30 different ASs in 24 different countries. The first 80 traces belong to 67 different ASs and 26 countries. This highlights the importance of utilizing vantage points that are geographically diverse and are hosted in different ASs.

To better understand both the need for diversity in vantage points as well as the underlying reasons behind the limited additional network coverage of hosting infrastructures by each trace, we perform a direct comparison of the traces. For

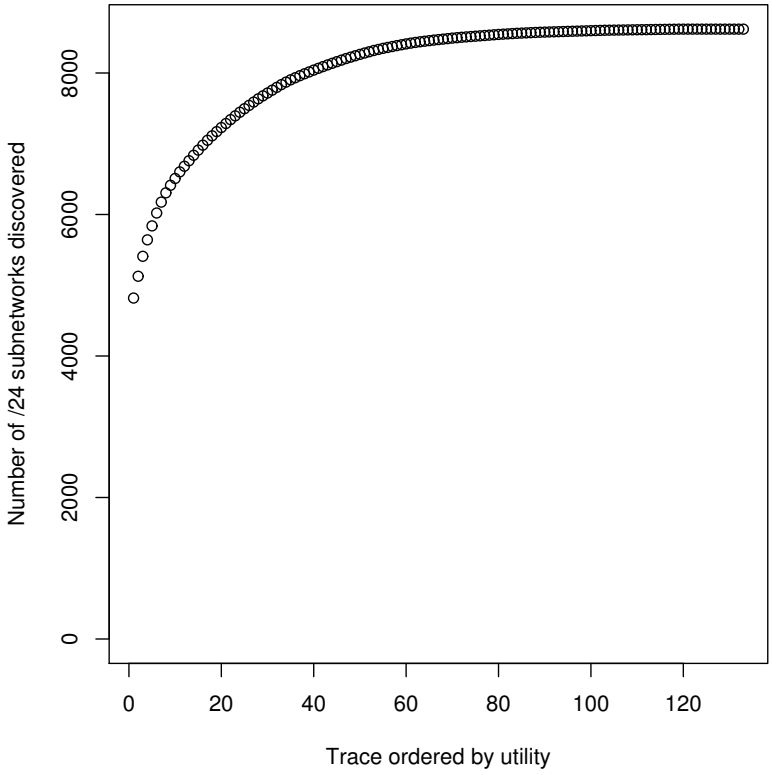


Figure 5.3: /24 subnetwork coverage by traces.

this, we define the similarity between two sets s_1 and s_2 as follows:

$$similarity(s_1, s_2) = 2 \cdot \frac{|s_1 \cap s_2|}{|s_1| + |s_2|} \tag{5.1}$$

where $|.|$ denotes the size of the set.

We define the /24 subnetwork similarity between two DNS replies for the same hostname as the similarity between their respective sets of /24 subnetworks. We define the similarity between two traces as the average of the /24 subnetworks similarities across all hostnames.

In Figure 5.4 we show the cumulative distribution of the similarity across all pairs of traces. We also show the similarity across traces when considering only one of the three subsets of the hostname list. The high baseline value of similarity highlights the need for diversity to sample hosting infrastructures. It also confirms the slow increase in the utility of the traces shown in Figure 5.3.

Let us now compare the curves for the different types of hostnames in Figure 5.4. We see that the similarity for TAIL2000 is very high, indicating the limited diversity in the location for the corresponding hosting infrastructure. This is contrasted with the similarity for EMBEDDED, that is the lowest among the four

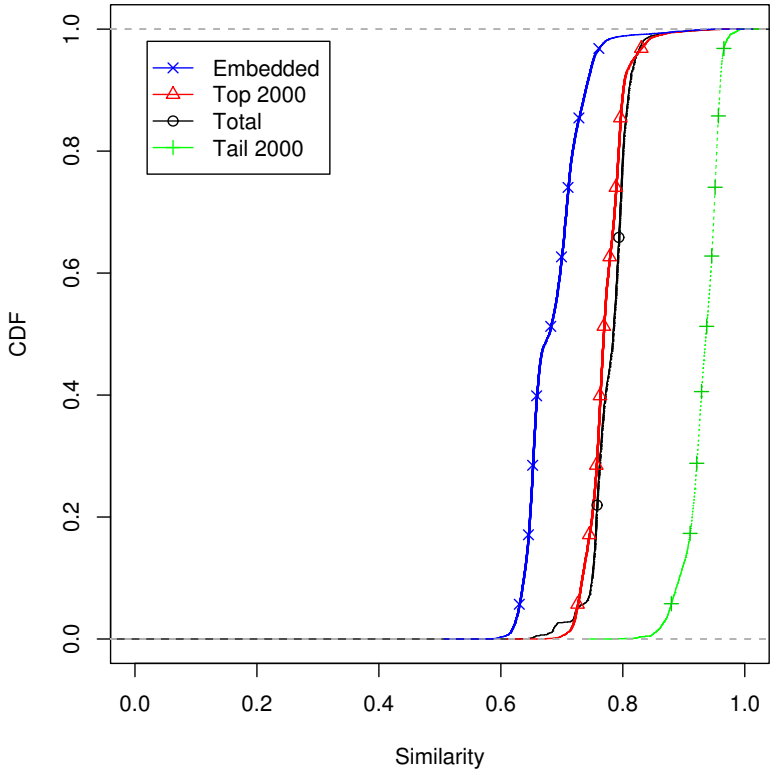


Figure 5.4: CDF of similarities for answers across different traces, for different sets of hostnames.

curves. A low similarity for EMBEDDED is the consequence of the nature of the corresponding objects: typically they have a long lifetime and often are large. This makes them prime candidates for being hosted on distributed infrastructures, e. g., CDNs. TOP2000 lies in-between TAIL2000 and EMBEDDED. This indicates that the corresponding hostnames are hosted on a mix of centralized and distributed hosting infrastructures.

5.4.4 Summary

The validation of our approach highlights the need for diversity in geographic and network location. Our choice of a mix of different hostnames, including popular, unpopular, and embedded objects, enables us to estimate the impact of our hostname list on the underlying networks that are utilized by hosting infrastructures. Popular hostnames and embedded objects seem to contribute the most in finding networks that are utilized by hosting infrastructures. Thus we believe that we are able to identify a significant part of such networks with our hostname list. Of course adding more hostnames may reveal other infrastructures. How-

ever, our coverage of popular and embedded hostnames is unlikely to miss large infrastructures. A larger number of vantage points is likely to increase the density of the sampling hosting infrastructures. However, for the aim of our study, this is of limited value given our current coverage of these distributed hosting infrastructures.

5.5 A Continent-level View of Web Content

Before delving into identifying characteristics of hosting infrastructures, we want to understand which parts of the world serve Web content. In this section we rely on the granularity of a continent. We quantify to which degree a user can find content in its own continent or has to obtain it from a different continent. This provides a view on the relative importance of different continents for Web content delivery as well the degree of replication of content.

In this section, we have to make sure to avoid bias due to the limited number of locations of third-party resolvers. Both GoogleDNS and OpenDNS have a limited number of locations. Moreover, during data collection, OpenDNS was only available in the US and Europe, but not in Africa, Asia or Oceania. Therefore we limit ourselves to *LocalDNS*.

5.5.1 Geographic Replication of Content

In this section, we examine the relationship between the locations of content requester and content location as identified by DNS answers. Each line of Table 5.2 summarizes requests that originate from a given continent. Columns of Table 5.2 break down the requests among the continents from which the requested content is served. Each line adds up to a 100 %, while columns do not as they reflect the global importance of a continent. The shade of each entry of Table 5.2 is a visual aid, directly indicating its value.

The three most prevalent continents in terms of served content are North America, Europe, and Asia. At least 42 % of the content is served from North America, 22 % from Europe and 19 % from Asia. The other three continents, namely Africa, Oceania, and South America, do not appear to serve popular Web content.

The second observation from Table 5.2 is a strong diagonal in the matrix, indicating that at least part of the content is fetched from the same continent. Subtracting the minimum of a column from the corresponding element in the diagonal reveals that up to 12.5 % of the content requests that can be served from multiple continents are served from the continent of the requester. This locality of Web content availability provides evidence that a considerable fraction of content is replicated in different regions of the world. Note, by choosing the granularity of continents, the existing diversity within continents is hidden. In addition, we observe an almost identical behavior for content requested from Africa and Europe.

We attribute this to the fact that Internet connectivity in Africa is mostly provided via Europe and the lack of local content replication infrastructure. Note, the observed behavior might also be due to a limited number of traces from Africa. Oceania and Asia localize to a lesser degree than either Europe or North America. Asia fetches the remaining content mostly from North America.

Table 5.2: Content matrix for the content mix. Each line provides the percentage of all requests that originate from a given content. Columns indicate the continent from where content is served.

	Served from					
Requested from	Africa	Asia	Europe	N. America	Oceania	S. America
Africa	0.3	19.7	34.4	42.2	0.9	0.7
Asia	0.3	27.5	22.3	45.4	1.3	0.7
Europe	0.3	19.7	34.6	42.1	0.9	0.7
N. America	0.3	19.6	22.4	54.6	0.9	0.7
Oceania	0.3	22.2	22.4	44.5	8.1	0.8
S. America	0.3	19.5	22.2	44.5	0.9	11.3

5.5.2 Content-dependent Replication

Content varies in both popularity and type. This is the reason why we distinguish not only popular and less popular hostnames, but also different types of embedded objects (see Section 5.3). In this section, we refine the previous analysis of the relationships between the locations of content requester and content origin by slicing the content matrix across our three subsets of hostnames: TOP2000, TAIL2000, and EMBEDDED.

When comparing the matrix of EMBEDDED (Table 5.3) with all the others (TOP2000 and TAIL2000 not shown, total in Table 5.2), we observe that the diagonal is more pronounced for EMBEDDED. This indicates that embedded objects are, on a continent-level, more locally available than content from the other sets. This is consistent with our earlier observation that embedded objects have a high utility in terms of IP discovery, cf. Section 5.4.2. We notice that Asia appears stronger for EMBEDDED compared to TOP2000 and TAIL2000. The matrices for TOP2000 and TAIL2000 (not shown) resemble the overall one (Table 5.2), except for a slightly higher amount of content delivered from North America at the expense of mostly Europe and Asia.

5.5.3 Summary

In this section, we analyzed the relative weights of Web content in different continents. We showed the prevalence of North America, Europe, and Asia in Web

Table 5.3: Content matrix for EMBEDDED. Each line provides the percentage of all requests that originate from a given content. Columns indicate the continent from where content is served.

	Served from					
Requested from	Africa	Asia	Europe	N. America	Oceania	S. America
Africa	0.3	26.9	35.5	35.8	0.3	0.6
Asia	0.3	37.9	18.3	40.1	1.1	0.6
Europe	0.3	26.8	35.6	35.6	0.4	0.6
N. America	0.3	26.5	18.4	52.9	0.3	0.6
Oceania	0.3	29.2	18.5	38.7	11.3	0.6
S. America	0.3	26.4	18.2	39.3	0.3	14.2

content presence, and how each region relies on each other. We observed a considerable local availability of content in most continents, implying that a considerable fraction of content is replicated across multiple continents.

5.6 A Portrait of Hosting Infrastructures

Given the multiplicity of ways to assemble hosting infrastructures, and the fact that new ones are continuously being deployed, we want to be able to understand the state of their deployment. Therefore, we turn our attention to the independent hosting infrastructures that are serving the hostnames from our list. In this section we identify the prominent hosting infrastructures, detect where they are actually located by ASs and countries, and classify them according to their network location footprint. Moreover, we study the geographic properties of hosting infrastructures and provide a ranking of countries according to their capability of serving popular Web content. In this section we strive for a good location coverage of hosting infrastructures, thus we choose *AllDNS* as input data.

5.6.1 Identifying Hosting Infrastructures

To distinguish between individual hosting infrastructures, we need to select features that allow us to differentiate between them. In this chapter, we use the following observed features: number of IP addresses, number of /24 subnetworks, number of ASs. These three features relate closely to typical deployment strategies of hosting infrastructures [53]. For example, a small data-center will be located within a single AS, have a limited number of /24 subnetworks, and a large number of IP addresses. A data-center-based CDN may rely on multiple ASs. A massively distributed CDN will rely on a large number of ASs.

Clustering Algorithm

The main goal of our classification is to put together all hostnames that are served by the same hosting infrastructure. Our assumption is that a hostname is served by a single hosting infrastructure. We are aware of a very small number of counter-examples, e. g., `s.meebocdn.net` that is served by both Akamai and Limelight. By the design of the algorithm we propose, these hostnames will be considered as being served by a hosting infrastructure of its own. In the rest of the section we present our algorithm that identifies hosting infrastructures based on our collected traces.

We choose a two-pass algorithm. During the first pass, we ensure that the prominent hosting infrastructures are identified. It also gives a higher bound on the size of the clusters. This prevents the second step from clustering small infrastructures with large ones. This may happen quite often because highly distributed hosting infrastructures may share address space with other hosting infrastructures.

Step 1: Separating the large hosting infrastructures The goal of our first step is to separate the large hosting infrastructures from the rest. We rely on the k-means algorithm [55] with, as input, the three features mentioned above for each hostname of our list. With this input, the k-means algorithm partitions the hostnames in up to k clusters in the feature space. We call these clusters k-means clusters. The clusters whose features have high values, i. e., in terms of number of IP addresses, number of /24 subnetworks, number of ASs, relate to widely-deployed infrastructures. On the other hand, smaller infrastructures that use very few /24 subnetworks and IP addresses are not sufficiently different, and therefore, can be found in the same cluster. Increasing the value of k in the clustering algorithm does not help, as the feature space simply does not allow to differentiate them.

Step 2: Distinguishing the small hosting infrastructures The goal of the second step is to build sub-clusters within each k-means cluster by finding the hostnames that are hosted on similar network locations in the IP address space. Therefore, we introduce a new feature: the set of BGP prefixes the hostname maps to. Based on the similarity between the sets of prefixes of two similarity-clusters, we decide if they belong to the same hosting infrastructure, and if so we merge these clusters. For this we re-use the similarity concept defined by Equation 5.1.

We initialize the second step of the algorithm by putting each hostname into its own cluster, called a similarity-cluster. For each k-means cluster, we perform a pairwise comparison of its similarity-clusters and merge them according to their similarity. We iterate the process until convergence to a fixed point. At this stage, each similarity-cluster identifies all hostnames used by a single content delivery infrastructure.

Tuning By design, our approach avoids clustering infrastructures that do not belong together. This is achieved by choosing appropriate parameters. We argue that the analysis in the following sections is quite robust against too fine clustering.

Choosing k too high will lead to clustering large hosting infrastructures into smaller clusters, while choosing it too low may result in a large overlap between hosting infrastructures. We select $k = 30$, and the whole interval $20 \leq k \leq 40$ provides comparable results according to our verification. For the second phase of the algorithm we used a merging threshold of 0.7 on the similarity between two similarity-clusters.

Clustering Results

As first step, we validate the accuracy of the algorithm. For this, we rely on ground truth information about specific hosting infrastructures, i. e., Akamai and Limelight. The ground truth in the case of Akamai consists of the names present in the `A` records at the end of the CNAME chain inside DNS replies, which follow typical patterns. In the case of Limelight, we can use the same approach, and in addition verify that IP addresses belong to the unique AS number of Limelight. Table 5.4 shows the top 20 clusters as identified by our algorithm. Checking the top 20 clusters leads to 0 false positives, while the ground truth for both Akamai and Limelight yield a few hostnames that are not included in top clusters, which have been separated in the first step due to their unusual features. Those hostnames are typically hosted on a very small subset of the whole infrastructure. We conjecture that these hostnames are intentionally treated differently.

The output of the algorithm leads to the identification of more than 3000 potentially individual hosting infrastructures. We provide on Figure 5.5, for each hosting infrastructure cluster, the number of hostnames from our list that are served by the hosting infrastructure of that cluster, on a log-log scale. Hosting Infrastructure clusters are ranked in decreasing order of hostname count. We observe that a few hosting infrastructure clusters are serving a large number of hostnames. Most of the hosting infrastructure clusters serve a single hostname. Such hosting infrastructure clusters that serve a single hostname each have their own BGP prefix. We infer from this that these are likely to be located in a single facility and in most cases serve non-replicated content. The top 10 largest content infrastructures clusters, which overall amount to less than 1 % of all clusters, are serving more than 15 % of the hostnames from our list. The top 20 are serving about 20 % of the hostnames.

The resulting clustering allows us to make qualitative observations, namely that well-known hosting infrastructures are represented. Table 5.4 lists the top 20 clusters in terms of the number of hostnames served by them from our list. Among them, we find well distributed CDNs such as Akamai, “hyper-giants” such as Google, and data-centers, such as ThePlanet. As can be seen from Ta-

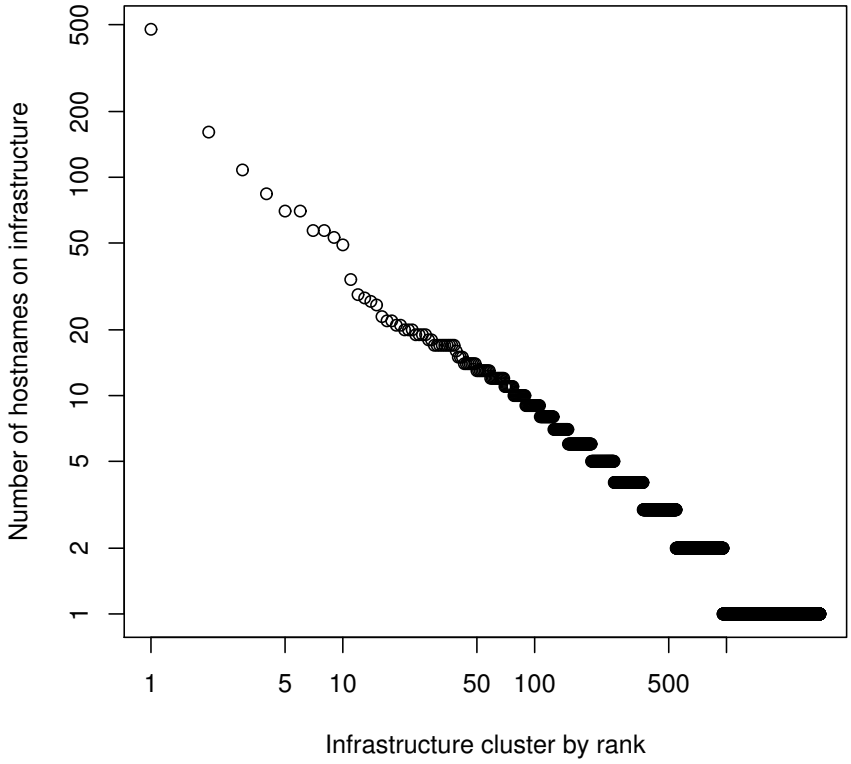


Figure 5.5: Hostnames served by different content infrastructure clusters.

ble 5.4, we find multiple hosting infrastructure clusters run by the same infrastructure authority. The explanation lies partly in the different types of services hosted by hosting infrastructures, as well as the geographic disparity of the infrastructure installation.

5.6.2 Classifying Hosting Infrastructures

Having identified the individual hosting infrastructures we turn our attention to quantifying the types of their deployment strategies. We concentrate on their features, especially the number of ASs and prefixes that a given cluster relies on. Table 5.4 also shows the number of ASs and prefixes for each of the top 20 content infrastructure clusters. Four natural patterns can be noticed from the <#ASs,#prefixes> tuples:

- A <very high,very high> tuple shows evidence of a massively distributed CDN, e. g., Akamai.

Table 5.4: Top 20 hosting infrastructure clusters by hostname count.

Rank	# hostnames	# ASs	# prefixes	Owner
1	476	79	294	Akamai
2	161	70	216	Akamai
3	108	1	45	Google
4	70	35	137	Akamai
5	70	1	45	Google
6	57	6	15	Limelight
7	57	1	1	ThePlanet
8	53	1	1	ThePlanet
9	49	34	123	Akamai
10	34	1	2	Skyrock
11	29	6	17	Cotendo
12	28	4	5	Wordpress
13	27	6	21	Footprint
14	26	1	1	Ravand
15	23	1	1	Xanga
16	22	1	4	Edgecast
17	22	1	1	ThePlanet
18	21	1	1	ivwbox.de
19	21	1	5	AOL
20	20	1	1	LeaseWeb

- A <high,high> tuple shows evidence of a CDN that is present in a few locations, e. g., Limelight, Cotendo, and Footprint.

- A <1,high> tuple shows evidence of a hyper-giant, e. g., Google.

- A <few,few> tuple shows evidence a possibly multi-homed data-center, e. g., ThePlanet, LeaseWeb, or simple Web hosting service.

In Section 5.6.3 we discuss how is it possible to quantify the values for "few" and "high".

To verify that the previous natural patterns can also be found among the more than 3,000 content infrastructure clusters, we show in Figure 5.6 a scatter plot of the number of ASs against the number of prefixes for all clusters. On the top right part of Figure 5.6, we find a number of clusters that correspond to massively distributed CDNs. In the center of the figure, we find less distributed CDNs. In the lower left corner, we find both hyper-giants and as we move closer to the origin, we find a large number of small data-centers.

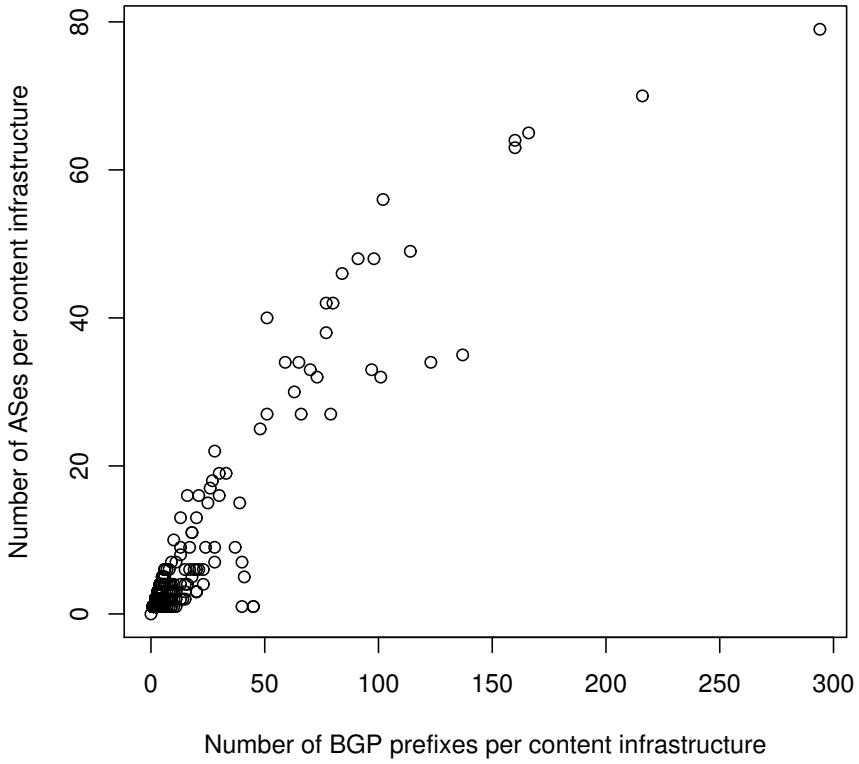


Figure 5.6: AS and prefix features for each content infrastructure cluster.

5.6.3 Geographic Properties of Content Infrastructures

The clustering of the content infrastructure presented in the previous section is agnostic with respect to geographic locations. To provide insight about the geographic properties of the different clusters we map the clusters to the geographic locations of their prefixes. A note of caution is obviously required: IP geolocation databases provide only limited geographic resolution, but have been shown to still be accurate at the country-level [87]. We rely on the Maxmind geolocation database [58] to map prefixes to geographic locations.

Distinguishing between content infrastructures that rely on a few ASs or prefixes is tricky, especially because we do not have a priori knowledge about their signature in terms of ASs and prefixes. Indeed, some of these clusters might very well be present in a single location but for administrative reasons split their infrastructure into multiple ASs or use multiple prefixes due to multi-homing. One known example is Rapidshare [3], that relies on multiple ASs and prefixes yet whose facility is a single data-center.

Therefore, for each content infrastructure cluster, we estimate based on geolocation information, the number of countries in which the cluster is present. For each cluster we check onto how many countries it is deployed. Figure 5.7 shows the resulting relationship in the form of a stacked bar-plot. On the x-axis, we show the number of ASs that host hosting infrastructure. We also provide in parenthesis the number of different clusters that have a given number of ASs. On the y-axis, we show the fraction of clusters whose prefixes are located in a given number of countries (see legend).

As shown by Figure 5.7, most of the hosting infrastructure clusters that use a single AS are present in a single country. As a cluster relies on more ASs, the likelihood that it is present in multiple countries increases. At the same time, a non-negligible fraction of hosting infrastructure clusters using multiple ASs are located in a single country. Because of the limited number of content infrastructure clusters with strictly more than 5 up to 10 ASs (24 clusters), the fraction for these in Figure 5.7 simply reflects a few instances of particular content infrastructures. Most of these clusters are present in several countries, so are probably CDNs. In addition, we see 37 clusters available in more than 10 ASes, of which 12 are available from 3 or more but less than 10 countries, and 16 are available from at least 15 up to 25 different countries.

5.6.4 Summary

In this section, we propose a clustering algorithm that is able to identify content infrastructures based on simple features. We validate our classification based on ground truth information for two large CDNs. We distinguish between smaller hosting infrastructures based on both their features and their geographic properties. We find a relationship between the number of ASs on which a hosting infrastructure relies on and the multiplicity of its locations, giving a hint about their deployment strategy.

5.7 Mapping Content Infrastructures

Having identified the different hosting infrastructures that serve popular Web content, we want to find out how they relate to each other. We examine their installation on a geographic-level as well as the AS-level. The geographic level tells us the hot-spots of content hosting in the physical world. The AS abstraction reveals how the geographic installation of hosting infrastructures maps to the overlay of networks in the Internet. Similar to Section 5.5 we avoid bias towards geographic locations or network locations by limiting ourselves to *LocalDNS*.

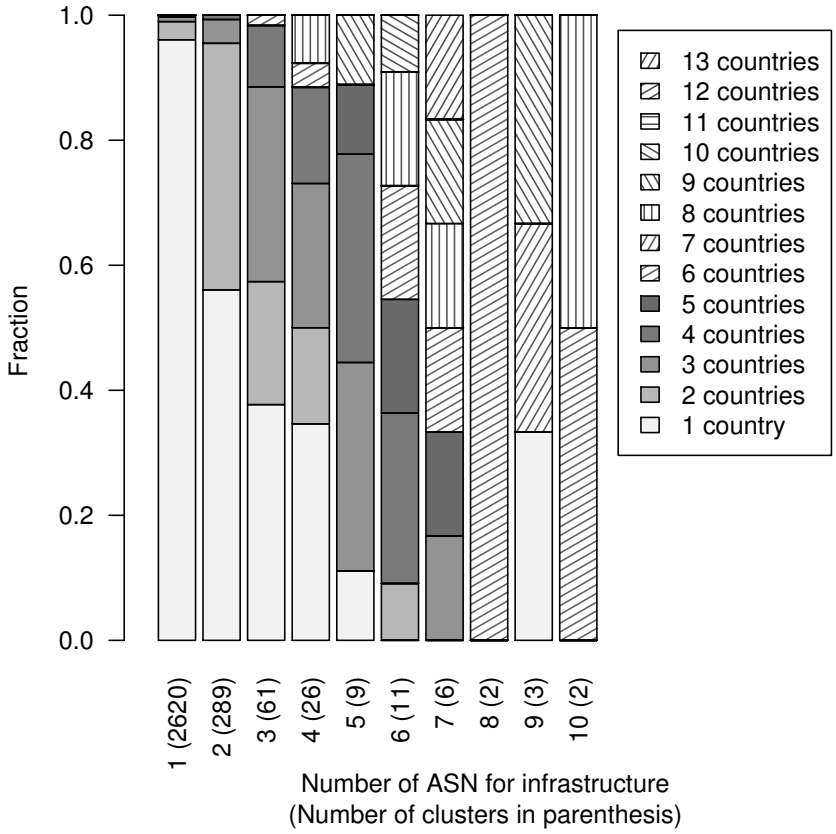


Figure 5.7: Country-level diversity of content infrastructure clusters.

5.7.1 Content Potential

To be able to compare to which degree different locations are able to serve content, we introduce the notion of *content delivery potential*. The content delivery potential is the fraction of hostnames that can be served from a single "location". Note, by location we mean a country or AS. The advantage of the content delivery potential is that it allows one to directly compare the fraction of hostnames that can be served by different locations. The disadvantage is that replicated content is counted as many times as there are different locations where it is hosted, introducing a bias in favor of replicated content.

Therefore, we also introduce the notion of *normalized content delivery potential*. The normalized content delivery potential divides the weight of a given hostname by the total number of locations it can be served from. Irrespective of the number of locations where a given hostname is hosted, its total weight is normalized to 1. Note, the normalized content delivery potential is additive, i. e., its value can be directly added across different locations to gauge the potential of a coarser region.

Table 5.5: Geographic distribution of content infrastructure, ranked by the normalized potential.

Rank	Country	Potential	Normalized potential
1	USA (CA)	0.254	0.108
2	China	0.128	0.107
3	USA (TX)	0.190	0.061
4	Germany	0.183	0.058
5	Japan	0.163	0.051
6	France	0.146	0.034
7	Great Britain	0.157	0.030
8	Netherlands	0.144	0.029
9	USA (WA)	0.135	0.027
10	USA (unknown)	0.164	0.027
11	Russia	0.038	0.027
12	USA (NY)	0.130	0.026
13	Italy	0.122	0.018
14	USA (NJ)	0.125	0.016
15	Canada	0.028	0.015
16	USA (IL)	0.116	0.014
17	Australia	0.118	0.013
18	Spain	0.116	0.013
19	USA (UT)	0.111	0.012
20	USA (CO)	0.113	0.012

5.7.2 Geographic Content Hot-spots

To find out the geographic location of the hot-spots that serve most hostnames from our list, we compute both content potentials on a per-country basis. Table 5.5 shows the results for both potentials. Note, for the USA only, we provide the state level. The lines of Table 5.5 is ranked by decreasing normalized content delivery potential and shows the Top 20.

Despite the division into states, the USA leads the ranking with its hosting infrastructure in California. Indeed, in total 9 US states are among the top 20. On the second place we find China. Directly comparing California with China reveals that China's delivery potential is a lot lower than California's, yet the values of their normalized potentials are quite close. Comparing China's potential with its normalized potential indicates that a large fraction of the content served in China is only available from China. In total, China and California together count for over 23 % of hostnames of our list in the normalized potential. Besides USA and China, 7 European countries are among the top 20, as well as Japan, Australia and Canada. In total we see content being delivered from 122 countries/US states, or 77 countries. The top 20 presented here are responsible for 70 % of all hostnames

in our study.

5.7.3 AS-level Content Hot-spots

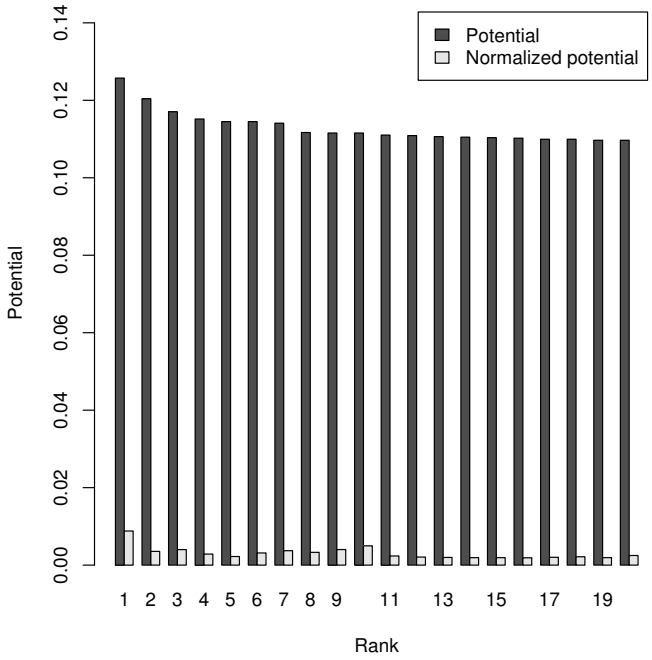
Geographic hot-spots are insightful in that they reflect where large chunks of the hosting infrastructure are. However, they provide little insight to understand how content is delivered to Internet users. Therefore, we investigate where content resides at the AS-level.

To map hosting infrastructure clusters to ASs, we rely on the same approach as in Section 5.6.3. For each cluster, we take the prefixes from which it serves content, and map each prefix to an AS number using BGP data. This gives us a set of AS numbers for each cluster. Recall that a content infrastructure cluster is defined by a set of hostnames it serves. We reuse the notion of content delivery potential, as introduced in Section 5.7.1, but where "locations" are now ASs. The content delivery potential of an AS is the fraction of hostnames it can potentially serve from all the clusters that are hosted on this AS.

Figure 5.8 provides the top 20 ASs in terms of their content delivery potential. Unexpectedly, we find mostly ISPs in this top 20. The two genuine content hosters in the list are Akamai and Bandcon. There are two main factors explaining the unexpected top 20: *(i)* all these ASs host Akamai caches that boosts their content delivery potential and *(ii)* all these ASs host some content that no other AS can provide. Given the widespread installation of Akamai caches in carriers, the second factor is actually more important, explaining why some ASs appear among the top and why others do not. The take-home message is that a content-driven AS ranking has to be normalized by content hosted on massively distributed hosting infrastructures.

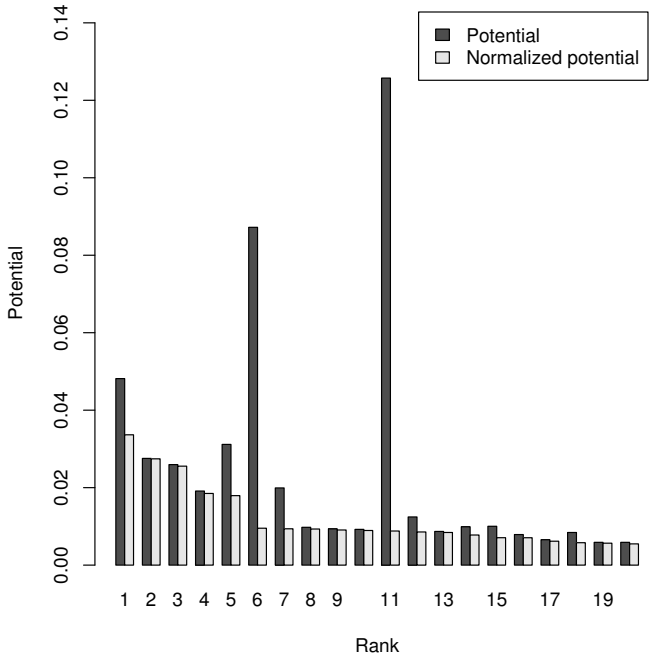
The normalized content delivery potential does exactly this and thus provides a more balanced AS ranking. It spreads the weight of distributed content infrastructure across all ASs that serve their hosted content. Figure 5.9 provides the top 20 ASs in terms of normalized content delivery potential. Our first observation is that the only overlap with the non-normalized ranking is NTT which was top ranked. The ASs that appear on the top of the normalized ranking do so because of the exclusiveness of the content they host. In other terms, the ranking reflects the monopoly some ASs have over popular content. As expected, Google is among the top ranked ASs due to its importance in popular content. We also see data-center content infrastructures: ThePlanet, SoftLayer, Rackspace, 1&1 Internet, OVH, Amazon, LeaseWeb, and Hetzner Online. A limited number of ISPs in China seem to also host a considerable fraction of popular content exclusively.

Content Monopoly Index If we look more closely at Figure 5.9, it is possible to spot some unusually high values of the non-normalized potential. These ASs are well-known tier-1 carriers. We also observe that in most cases, the value of the



Rank	AS name
1	NTT America
2	Tinet
3	Global Crossing
4	KDDI
5	Akamai Europe
6	Telia
7	Deutsche Telekom
8	Korea Telecom
9	Qwest
10	Bandcon
11	Cable and Wireless
12	SingTel
13	Akamai
14	France Telecom - Orange
15	Internode
16	Comcast
17	StarHub
18	nLayer
19	Beyond The Network
20	TATA

Figure 5.8: Top 20 ASs in content delivery potential.



Rank	AS name	CMI
1	Chinanet	0.698
2	Google	0.995
3	ThePlanet	0.984
4	SoftLayer	0.967
5	China169 Backbone	0.575
6	Level 3	0.109
7	China Telecom	0.470
8	Rackspace	0.954
9	1&1 Internet	0.969
10	OVH	0.968
11	NTT America .	0.070
12	EdgeCast	0.688
13	GoDaddy.com	0.969
14	Savvis	0.785
15	China169 Beijing Province	0.705
16	Amazon.com	0.895
17	LEASEWEB	0.942
18	Cogent	0.686
19	Hetzner Online	0.962
20	AOL	0.931

Figure 5.9: Top 20 ASs in normalized content delivery potential.

non-normalized potential is close to the one of the normalized potential. These ASs with close values of their normalized and non-normalized potential correspond to data centers. To facilitate the comparison of ASs into those that have exclusive content with ASs that host replicated content, we introduce the *content monopoly index* (CMI), which we define as the ratio between the normalized content potential and the non-normalized content potential. An AS with a large CMI hosts a large number of hostnames not available elsewhere, as compared to content that is replicated in other ASs.

5.7.4 Content vs. Traditional AS Rankings

Given the multiple of AS rankings that have been introduced in the literature [7,49,76], we compare content-based rankings to the traditional ones. One of the most well-known AS ranking is the one from CAIDA [7]. CAIDA proposes two different types of rankings: one based on AS-degree and another on the size of the customer cone of an AS. The size of the customer cone of an AS is the set of ASs, IPv4 prefixes, or IPv4 addresses that can be reached from a given AS following only customer links. The larger the customer cone of an AS, the more important this AS is deemed to be. The cone-based classification assumes that an AS that has a larger fraction of the ASs, prefixes or IP address space will attract more traffic. Labovitz et al. [49] question this assumption and provide a different AS ranking based on the amount of traffic that an AS serves.

Table 5.6 compares 6 different AS rankings: the CAIDA AS-degree (CAIDA-degree) and customer cone (CAIDA-cone) rankings [7], a ranking similar to CAIDA's by Renesys (Renesys) [76], the traffic exchanges-based ranking by Labovitz et al. [50][3] (Arbor), and finally our content-based rankings (potential and normalized potential) as defined above. When observing the purely topological rankings such as the ones from CAIDA and Renesys, we observe that they tend to rank large transit carriers high. Besides the case of Google and Comcast, the top of Arbor's ranking leads to similar results to topological rankings. Our content infrastructure-driven rankings on the other hand give more weight to those ASs that deliver a large amount of content. We notice that our normalized potential leads to similar top ranked ASs as topological and traffic-based rankings, while of course favoring ASs that host content.

We insist on the fact that no AS ranking captures all relevant aspects of the importance of an AS. All aspects of the Internet are important, i. e., topology, traffic, and content, and need to be taken into consideration to understand the Internet ecosystem.

[3] Some of the entries of the Arbor ranking were intentionally omitted by [49].

5.7.5 Summary

We proposed two different ways to rank ASs based on their content potential. We showed that these rankings reveal different aspects of content hosted by ASs: replicated content and content exclusively hosted by a given AS. We proposed an index, called the content monopoly index, which measures the degree to which an AS hosts content not available elsewhere, compared to content that is replicated in other ASs. Finally, we related our content-centric rankings to those presented in the literature.

5.8 Discussion

When we introduce our content-centric hosting potentials of ASs in Section 5.7.1, we omitted to discuss their implications regarding the recent shifts in traffic observed by Labovitz et al. [49]. They observed the emergence of a new class of important ASs, coined "hyper-giants". Our rankings also identify such ASs as dominant from a content-centric perspective. However, two factors limit the view of both Labovitz et al. [49] as well as topology-driven rankings. First, topology-driven rankings underestimate the power of stub ASs in terms of content delivery. We find that many of the top ASs in our content-centric ranking are stub ASs, including large data-centers such as Google, ThePlanet and Amazon. These stub ASs rely on tier-1 ASs for Internet-wide content delivery, as well as direct peerings with a limited number of regional providers. Moreover, as observed by Labovitz et al. [49], the relative importance of those direct peerings has increased at the expense of tier-1 ASs.

Second, traffic-driven rankings underestimate the impact of content replication because this traffic is served locally, hence is not visible, e. g., to Labovitz et al. [49]. Our normalized content potential (Section 5.7.1) pinpoints that phenomenon. To conclude, we highlight the finding of Labovitz et al. [49] who observe a shift in the Internet AS ecosystem, away from large carriers and toward content providers. Our work explains the nature of this shift: the power of content.

5.9 Related Work

Recent studies provide evidence in support of the significant rise of Web content traffic [49,56]. Two major reasons are the growth of video traffic and the increasing penetration of broadband access. To cope with these changes, large-scale content distribution networks are being deployed [29, 46, 53]. In addition, applications that used to rely on peer-to-peer delivery such as file sharing are nowadays increasingly served from data centers, or One-click Hosters [3, 49].

Labovitz et al. [49] observed consolidation of Web content traffic as well as a significant shift in peerings to better facilitate connectivity to content providers.

They analyzed inter-domain traffic over a two year period, relying on data from 110 commercial ISPs and content providers. They detect global shifts in interdomain traffic and the AS ecosystem.

Huang et al. [37] and Su et al. [90] leverage DNS requests to understand the distribution of content. However, their work is restricted to the study of specific CDNs, and does not try to detect different types of hosting infrastructures. Utilizing DNS replies of popular content in order to identify the location of hosting infrastructures shares similarities with work by Krishnamurthy et al. [45]. Yet, their focus was on studying the performance of DNS resolvers, rather than leveraging DNS for Web content cartography.

Researchers in the past proposed models to estimate inter-AS traffic matrices [8], or to estimate inter-domain Web traffic demand [25]. Chang et al. [8] rely on DNS answers for a large number of URLs that correspond to popular Google keywords, and the mapping of returned IPs to ASs, to classify ASs. To that end, they were able to classify ASs into three categories: Web service, residential, or business access. This serves as basis to generate an empirical model of inter-AS traffic matrices. Feldmann et al. [25] correlate requests to a large distributed CDN with those of third parties (e. g., embedded objects or advertisements) in order to estimate Web traffic demand.

Finally, note that studying P2P content is out of the scope of this chapter. Here we refer to work by Cuevas et al. [15] which shows that a small number of hosting infrastructures is responsible for the upload and download of a significant fraction of P2P content.

5.10 Summary

In this chapter, we introduce Web content cartography. Web content cartography aims at building maps of the Web content infrastructure. To demonstrate the benefits of content cartography, we build a map of Web content hosting and delivery infrastructures.

We build maps of the content infrastructure at multiple levels, e. g., continent, country, ASs. At each level, we analyze the Web hosting infrastructure. Our analysis highlights the prevalence of content replication in the Internet. We propose a novel approach to identify individual hosting infrastructures through their network footprint.

By sampling the facilities of hosting infrastructures at the AS-level, we are able to quantify the replication of content in ASs. We introduce content-centric AS rankings that quantify the ability of an AS to serve content. We compare our rankings with existing ones that rely on network topology and traffic.

Moreover, we introduce the content monopoly index that distinguishes between ASs that serve mostly replicated content and those, that host content exclusively served by them. Our observations support the findings of Labovitz et al. [49] and explain it through the power of content.

Table 5.6: Topology-driven AS rankings against traffic-driven and content-based AS rankings.

Rank	CAIDA-degree	CAIDA-cone	Renesys	Arbor	Potential	Normalized potential
1	Level 3	Level 3	Level 3	Level 3	NTT	Chinanet
2	Cogent/PSI	AT&T	Global Crossing	Global Crossing	Tinet	Google
3	AT&T	MCI	Sprint	Google	Global Crossing	ThePlanet
4	MCI	Cogent/PSI	NTT	*	Deutsche Telekom	SoftLayer
5	Hurricane	Global Crossing	Savvis	*	KDDI	China169 backbone
6	Qwest	Sprint	TeliaSonera	Comcast	Telia	Level 3
7	Sprint	Qwest	Tinet	*	Akamai	Rackspace
8	lobal Crossing	Hurricane Electric	Verizon	*	Bandcon	China Telecom
9	tw telecom	tw telecom	AT&T	*	Cable and Wireless	1&1 Internet
10	INIT7	TeliaNet	China Telecom	*	Qwest	OVH

6 Discussion

While predicting the future is hard, we nevertheless now speculate how the Internet may develop over the next few years, and possible consequences. We claim that a major factor for shaping the Internet landscape are the users. Their demand for content and their performance demands determine which business cases have the potential to be successful.

In this chapter, we argue, that client-server based content distribution technologies will stay dominant for quite some time because of their superior performance. We discuss design criteria for planning a content distribution infrastructure, and present recommendations to all parties—ISPs, content distribution infrastructures, and users—how to improve content delivery. We encourage to think about the roles of organizations in the Internet, and conclude with a list of open questions.

6.1 The Asymmetry in the Access Bandwidth and its Consequences for Content Delivery

Today's broadband providers mostly offer Internet access with highly asymmetric bandwiths [44]. For example, in Germany a typical ADSL product for residential users provides 16 Mbps downstream bandwith, but only 1 Mbps in the upstream, i. e., a factor of 16 difference.

This asymmetry in the access network bandwiths affects the performance of architectures that rely on the user's infrastructure for replicating content: Their total download capacity is limited by the total upload capacity of all participating peers. This concerns pure P2P systems as well as server-based hosting infrastructures on the residential user's premesis as recently proposed [60]. For the rest of the section, we will use the term *user-hosted infrastructure* to describe both architectures in a single phrase. The advantage of user-hosted infrastructure is that content can be pushed nearer to the consumer which promises network savings [51,95], and that the cost for the content provider can be reduced by utilizing the user's upstream bandwith for free.

These are considerable advantages, but how does the performance of user-hosted infrastructures compare to other content delivery infrastructures? To answer this question, we have to investigate how long it takes to fetch an object, i. e., the sum of the delays of finding a content server and for downloading the

content. Which of them is dominating is determined by the architecture in use and by the workload. Thus, we want to explore the possibilities and limitations of serving content from the user's systems with two typical use cases in mind: *(i)* Bulk downloads, i. e., a single large file is downloaded, and *(ii)* Web browsing, i. e., many small files that together compose a Web page need to be fetched at the same time.

For the first scenario, bulk downloads, we can in principle overcome the bandwith asymmetry of the uploading peers. It is a well known fact that the aggregated bandwith utilization of many users is much smaller then the sum of their link capacities [54, 66]. This is underlined in recent work by Maier et al. [56] and Siekenen et al. [86], who have shown that the average utilization of the provided bandwith is on average very low. This allows to develop protocols that use the aggregated upload bandwith of all users to saturate the download capacity of the comparatively few users that are downloading content at any given time. In addition, the overall download time is dominated by bandwith constraints: A few additional round trip times for finding the content will not dominate the overall performance.

Nevertheless, the asymmetry of the access link capacity is the bottleneck in real-world P2P implementations: there often are not enough uploaders participating to satisfy the demand of the downloaders. This is consistent with observations by Antoniades et al. [3], Maier et al. [56] and by ourselves [43], showing that the performance of P2P applications is about an order of magnitude lower than the performance of client-server based protocols. Still, this is an acceptable tradeoff for applications for which the total download time is secondary to the delivery cost, or where the download time can be reduced by using a hybrid model, e. g., the Blizzard downloader[1] for games and patches.

In the second scenario, Web browsing, most objects are small, and a single Web page consists of many such objects. Given the undercapacitated upstream bandwith objects need to be stored in a distributed fashion across all participating systems, and the replication degree of each object needs to increase with its popularity. There are two choices for finding the content: *(i)* a centralized directory or *(ii)* a distributed system such as a distributed hash table. The first option promises better performance: in principle the content lookup can be done in a single round trip time, while a distributed approach typcially results in lookup times that scale with the logarithm of the number of participating nodes. However, the centralized solution requires additional centralized hardware, the database entries need to be maintained, and it is only applicable for user-hosted servers as it would defeat a P2P approach.

In addition, it is not necessarily justified to expect content to be nearer to the client due to positive effects of caching, cf. Chapter 3. For example, in many cases the limited lifetime or even non-cacheability of a Web-object (Chapter 3) impairs

[1] `http://www.wowpedia.org/Blizzard_Downloader`

the potential gain from a user-hosted infrastructure.

Comparing user-hosted infrastructures with the state of the art in content delivery is not easy as the result depends on the workload. However, one difference is striking out: The network overhead for coordinating and maintaining content in the user-based infrastructure is higher than for a content delivery infrastructure based on DNS and HTTP.

It is a major research challange to find scalable solutions to these problems currently conducted in projects such as NADA [95] and, for streaming applications, P2P-Next [65]. While symmetric access lines could mitigate the problem to some degree and niche markets for P2P based content delivery exist, today server-based content delivery infratructures have the edge in terms of performance. This suggests, that server-based content delivery infrastructures will stay the most important sources of content for at least a few years.

6.2 Data Center vs. Highly Distributed CDN

In the last section we argued that server-based content delivery infrastructures will prevail for some time because they satisfy performance demands better than alternative approaches. This raises the question, how should an ideal content delivery infrastructure be designed.

Leighton discusses several performance, cost, and management aspects of CDNs and comes to the conclusion that highly distributed infrastructures are the most desirable despite the higher effort of deployment [53]: According to Leighton the reasons include better resilience against network problems, higher total link capacity available to the content infrastructure, cheaper hardware, energy and bandwith oftentimes being provided for free by partnering ISPs, and smaller latencies resulting in higher TCP throughput [53]. On the other hand, we showed that CDNs do not always make performance-optimal decisions when assigning their content servers [71]. Triukose et al. even claim that the highly distributed Akamai CDN could be consolidated into 60 or less data centers "without noticable performance penalty" [94]. Note, that the scope of view of Triukose et al. [94] may be limited by the use of PlanetLab as measurement platform.

In Section 5.6.1 we classified hosting infrastructures by their network footprint. The results indicate that, indeed, the infrastructures responsible for the highest fraction of hostnames are also the most distributed ones.

Yet, other considerations need to be taken into account when designing a hosting infrastructure. The type of application plays an important role. In case the application is latency sensitive, e. g., interactive completion of user input as seen on many Websites today, a highly distributed infrastructure has the better potential to achieve the user's satisfaction. Gaming is another such example. On the other side of the spectrum reside large downloads. One click hosters such as Rapidshare and Megaupload seem to achieve competitive performance even with

centralized infrastructure. An ad-company specialized on the German-speaking market, ivwbox.de, even hosts it's single massive data center over a single upstream provider. For certain kinds of services some centralization is actually required. For example, it would be very challenging to connect the front end servers of Amazon's Web shop with a coherent backend database in a reliable fashion if the front-end servers were highly distributed.

A factor that is still understudied in the context of content delivery infrastructures is the overall energy consumption of the system: servers, network, and clients. Feldmann et al. [24] found that a well-distributed CDN requires less energy than P2P or datacenter infrastructures for delivering IPTV streams. We agree with their argumentation of the network part becoming a more important factor for energy consumption in the future. Yet, we point out that the workload of a general purpose content infrastructure differs considerably from a streaming application such as IPTV. Given the poor cachability of a major fraction of the content (Chapter 3), only the strategic placement of content delivery hosts as opposed to general-purpose caches in the vacinity of the clients may bring relief to the network by lowering the bandwith requirements. However, it is not clear how far the edge caches can be pushed without sacrificing too much in terms of cacheability due to shrinking user population (cf. Section 3.4.3), server-side energy consumption due to badly utilized servers, and maintainability.

To sum up, from the performance, the network traffic, and the energy perspective, it seems desirable to have a highly distributed infrastructure. However, this is not always feasible, and there are intrinsic limits on how far the distribution can be pushed: communication requirements with backend servers, cacheability, server utilization, and maintainability.

6.3 Implications

In this section we give recommendations of how to improve the existing infrastructure, and encourage to consider a new way of thinking about the roles of organizations in the Internet.

6.3.1 Recommendations for Users and Content Infrastructures

Both Google Public DNS and OpenDNS market their products with enhanced performance as compared to the ISP provided DNS server [32, 63]. This perception is supported by tools such as Namebench [61] which allow a detailed performance analysis even for unexperienced users. However, this analysis can be misleading: choosing a third-party resolver will most likely result in the selection of content servers that have worse connectivity (Chapter 4) and therefore will reduce the overall performance of Web browsing. The data set examined in Chapter 5 also contains TCP pings towards the discovered IP addresses which we

can use to quantify the effect. For example, when comparing the average latencies towards the Akamai CDN caches, the distance of the client towards the server returned by the local resolver is on average 39 ms. The latency towards the CDN cache returned by Google Public DNS resp. OpenDNS is on average 67 ms resp. 68 ms, i. e., almost twice as high. Figure 6.1 shows a scatter plot providing more details. Thus, a recommendation for performance-sensitive users is to choose the DNS resolver provided by the ISP.

A content delivery infrastructure can mitigate this user-induced problem by using PaDIS [71] and/or the DNS client subnet extension [14]. However, neither are standardized so far, and both need support by the DNS resolver in use. For content infrastructures, collaboration with ISPs and operators of third-party resolvers seems prudent to find a general solution that satisfies the needs of both ISP and content infrastructure.

In addition, a particular point manifested in Chapter 3: the amendable usage of cache control in HTTP by the content providers, mostly for services offering user generated content. Being less restrictive would give ISPs more freedom in engineering their network while at the same time lowering the load on the content servers.

6.3.2 Recommendations for ISPs

The ISP's goal is to minimize expenses while still delivering good enough performance to attract customers. Caching appears to be a viable option to achieve both goals by reducing network load and enhancing the customer experience at the same time. Indeed, recent developments indicate that caching may be worthwhile again for certain setups. Companies such as Oversi[2] offer ISP grade caching solutions. However, our investigation (Chapter 3) shows that caching is still difficult and not necessarily beneficial. A careful evaluation of the tradeoffs appears prudent before re-introducing HTTP caches into a network. In particular, if servers of content delivery infrastructures are already placed inside or nearby the network the positive effect of dedicated caches may be minimal. Instead utilizing these servers in a smarter way may bring the desired benefit without major investment. We discuss this approach in parallel work and introduce the PaDIS framework [71].

DNS is a key component of today's content delivery. In Chapter 4 we show how much the choice of a third-party DNS resolver impacts the choice of the content delivery server: External DNS servers almost never return content hosts inside a client's ISP even if such hosts are in principle available. For a user this implies worse performance (Section 6.3.1), for an ISP this may result in higher expenses. We can only speculate on how many users choose a third-party DNS resolver over the one provided by their ISP. We are aware of two reference points: On a vantage

[2] http://www.oversi.com/

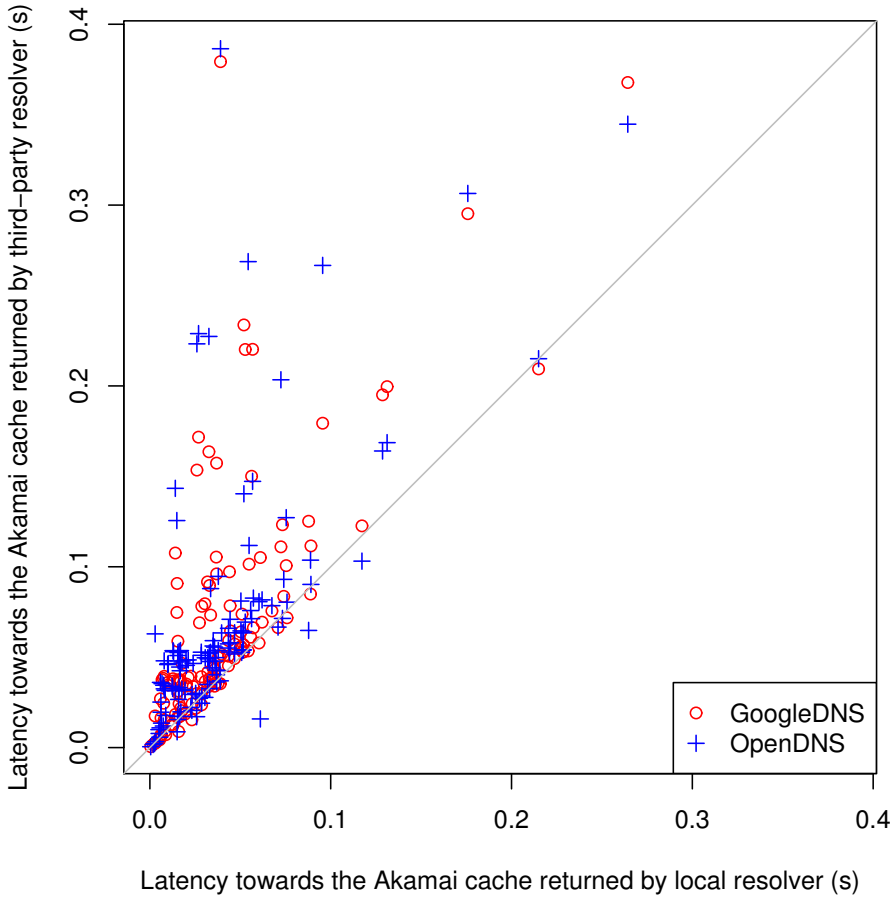


Figure 6.1: Comparing the latencies towards content servers returned by the local resolver with those returned by third-party resolvers. For each of the 133 vantage points we calculate the average latency towards the content cache returned by the local resolver, OpenDNS, or GoogleDNS, for more than 800 akamized hostnames. Each point in the scatter plot corresponds to a vantage point, and the coordinates represent the averaged latencies towards the content caches as returned by the local resolver (x-axis) and one of the third-party resolvers (y-axis). Points near the gray diagonal correspond to vantage points at which the performance is similar. In most cases, the local resolver returns IP addresses for content caches with lower latency.

point within a large European ISP (cf. Section 3.2) less then 1 % of the total number of DNS requests are targeted at third-party resolvers, and Kreibich et al. report that more than 12 % of the Netalyzer users use OpenDNS, but they assume a "geek-bias" in the user base [44]. An open question is why users choose a different DNS resolver than the one provided by the ISP. Reasons may include better DNS

performance of these servers (cf. Section 4.4.1), NXDOMAIN redirection or DNS blockages fostering distrust, or add-on services such as virus protection.

From an ISP perspective this means that ISPs need to keep their DNS infrastructure competitive. In particular we identified a number of Internet providers with highly centralized and/or load-balanced DNS infrastructure that performed worse than the respective third-party resolvers (Chapter 4). An approach where DNS resolvers are located nearer to the edge would improve the DNS performance and the server selection.

6.3.3 Web Content Cartography

We introduce a technique to estimate the content-centric hosting potentials of ASs in Chapter 5, that highlights the observations by Labovitz et al. [49] of the emergence of a new class of important ASs, coined "hyper-giants". However, two factors limit the view of both Labovitz et al. [49] as well as topology-driven rankings. First, topology-driven rankings underestimate the power of stub ASs in terms of content delivery. Second, traffic-driven rankings underestimate the impact of content replication because this traffic is served locally, hence is not visible, e. g., to Labovitz et al. [49].

Our work highlights the power of content: We find that many of the top ASs in our content-centric ranking are stub ASs, including large data-centers such as Google, ThePlanet and Amazon. These stub ASs rely on tier-1 ASs for Internet-wide content delivery, as well as direct peerings with a limited number of regional providers. Moreover, as observed by Labovitz et al. [49], the relative importance of those direct peerings has increased at the expense of tier-1 ASs.

To achieve more insight, the combination of different factors needs to be examined, e. g., content potential, content monopoly, content popularity, traffic volume, address space, size and mixture of user base, revenue, or profit. How much do these factors depend on each other? What makes an AS or organization important?

6.4 Open Questions

We already raised several technical and business-related questions that deserve further investigation in this chapter: How far can and should content caches be pushed towards the edge? How can ISPs and CDNs collaborate? What is the right way of handling client location in DNS? Why are users switching to third-party resolvers, and what can an ISP do to keep his DNS service attractive? What is the overall energy consumption of content delivery, and can it be reduced? How can the relevance of organizations in the Internet be quantified fairly?

A question that society will have to answer regards values: Currently, network neutrality and environmental protection are both highly debated topics in pol-

itics and well-reflected in media. Network neutrality proponents like to argue that a flat pricing model is required to keep the Internet free: freely accessible and freely useable for everyone. Proponents of environmental protection argue for reduction of our carbon footprint, in particular the reduction of our power consumption, or, in general, the thoughtful usage of natural resources. Even if it is not apparent on first sight, these are potentially conflicting goals.

Today, only a fraction of the content providers are using well-distributed content delivery infrastructures. Some even rely on a single data center. This results in a higher than necessary energy consumption (Section 6.2), while the current energy consumption of the Internet is already at an eminent level and predicted to grow [88]. From the environment viewpoint, we should create incentives for content providers to re-engineer the infrastructure in an energy and resources preserving way *at a global scale*. A straight-forward way to make a content provider aware of its resource and energy usage on the network is to let prices reflect the actual cost of transport: Long paths should be more expensive than short paths. This will motivate a content provider to also consider the network when planning a service.

However, path-length dependent pricing violates the concept of network neutrality. It is up to society to decide, which value it weighs higher: environmental protection or network neutrality.

7 Summary

In this thesis we characterize the impact of content delivery on the network. We study three aspects of content delivery: *(i)* how effective can caching be, *(ii)* what is the role of the server assignment mechanism and how does it interact with the user, and *(iii)* where from can content actually be fetched.

We find that P2P protocols are very well suited for caching when using the appropriate mechanisms. However, they only contribute a small fraction of the overall traffic volume. HTTP is responsible for the largest traffic share but is quite limited in its cacheability. NNTP is hardly cacheable at all.

Third-party resolvers, in general, do not manage to direct clients to content caches available inside the client's ISP, contrary to the ISP's DNS resolvers. Moreover, users of a surprisingly large number of ISPs suffer from poor latency towards their DNS resolvers. The DNS setup of serveral ISPs shows evidence of load balancing which results in reduced DNS cache utilization.

Using DNS observations in 78 autonomous systems we build maps of hosting infrastructures and highlight the prevalence of content replication. Next, we develop an automatable and lightweight technique to identify individual infrastructures based on their network footprint. This allows us to quantify to which degree content is replicated to different locations, e. g., ASs or countries. We define an index that reflects the degree of exclusively hosted content within an organization. Our findings provide a detailed analysis of the impact of locality on content delivery. We provide insight on the topological changes as they have recently been reported.

We discuss work related to our findings, and argue that client-server based hosting will stay the most important source of content for at least the next few years. We review general design criteria for content delivery infrastructures and ask if energy consumption should become a more central design objective. Based on our findings, we give recommendations to ISPs, content delivery infrastructures, and users on how to improve content delivery, and conclude with a list of open questions.

Acknowledgements

I am very grateful to my advisor Anja Feldmann. Anja helped me over many hurdles during the course of my PhD studies, showed confidence in my abilities during difficult times, supported and mentored me, and kept pushing me into the right direction.

I am extraordinarily grateful to Steve Uhlig who proved to be a very valueable co-advisor. I very much enjoyed the technical as well as the high-level discussions with him. These discussions helped me a lot in understanding the scientific "business". My gratitude also goes to Georgios Smaragdakis and his plenitude of ideas. Both Steve and Georgios provided excellent support and mentoring during the last one and a half years of my PhD studies, which were also the most productive ones.

Next, I want to thank Olaf Maennel and Matt Roughan for making my internship at the University of Adelaide possible. It was a wonderful experience at a beautiful place during which I learned a lot. Finally, my gratitude goes to my coauthors and friends Fabian Schneider, Wolfgang Mühlbauer, Ingmar Poese, and Benjamin Frank. We certainly learned from each other and had a lot of fun together.

List of Figures

List of Tables

Bibliography

[1] AGGARWAL, V., FELDMANN, A., AND SCHEIDELER, C. Can ISPs and P2P users cooperate for improved performance? *SIGCOMM Comput. Commun. Rev.*, 3 (2007).

[2] Alexa top sites. `http:///www.alexa.com/topsites`.

[3] ANTONIADES, D., MARKATOS, E., AND DOVROLIS, C. One-click Hosting Services: A File-Sharing Hideout. In *Proc. ACM Internet Measurement Conference* (2009).

[4] BERNERS-LEE, T. HyperText Transfer Protocol Design Issues. `http://www.w3.org/History/19921103-hypertext/hypertext/WWW/Protocols/DesignIssues.html`, 1992.

[5] BERNERS-LEE, T. Why a new protocol. `http://www.w3.org/History/19921103-hypertext/hypertext/WWW/Protocols/WhyHTTP.html`, 1992.

[6] Azureus messaging protocol. `http://www.azureuswiki.com/index.php/Azureus_messaging_protocol`, 2009.

[7] The CAIDA AS Ranking. `http://as-rank.caida.org/`.

[8] CHANG, H., JAMIN, S., MAO, M., AND WILLINGER, W. An Empirical Approach to Modeling Inter-AS Traffic Matrices. In *Proc. ACM Internet Measurement Conference* (2005).

[9] CHEN, X., AND HEIDEMANN, J. Flash crowd mitigation via adaptive admission control based on application-level observations. *ACM Trans. Internet Technol.* (2005).

[10] CHOFFNES, D. R., AND BUSTAMANTE, F. E. Taming the torrent: a practical approach to reducing cross-ISP traffic in peer-to-peer systems. In *Proc. ACM SIGCOMM* (2008).

[11] Cisco Visual Networking Index: Forecast and Methodology, 2009–2014. `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf`, 2010.

[12] CLAFFY, K. C., BRAUN, H., AND POLYZOS, G. Traffic Characteristics of the T1 NSFNET Backbone. In *Proc. IEEE INFOCOM* (1993).

[13] CLAFFY, K. C., AND BROWNLEE, N. Understanding Internet Traffic

Streams: Dragonflies and Tortoises. *IEEE Communications Magazine* (2002).

[14] CONTAVALLI, C., VAN DER GAAST, W., LEACH, S., AND RODDEN, D. Client subnet in DNS requests. IETF draft, work in progress, `draft-vandergaast-edns-client-subnet-00`, 2011.

[15] CUEVAS, R., KRYCZKA, M., CUEVAS, A., KAUNE, S., GUERRERO, C., AND REJAIE, R. Is content publishing in BitTorrent altruistic or profit-driven? In *Proc. ACM CoNEXT* (2010).

[16] DHAMDHERE, A., AND DOVROLIS, C. Ten Years in the Evolution of the Internet Ecosystem. In *Proc. ACM Internet Measurement Conference* (2008).

[17] Whois Source – Domain Counts & Internet Statistics. `http://www.whois.sc/internet-statistics/`.

[18] DREGER, H., FELDMANN, A., MAI, M., PAXSON, V., AND SOMMER, R. Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection. In *Proc. Usenix Security Symp.* (2006).

[19] ERMAN, J., GERBER, A., HAJIAGHAYI, M. T., PEI, D., AND SPATSCHECK, O. Network-aware Forward Caching. In *Proc. World Wide Web Conference* (2009), ACM.

[20] FANG, W., AND PETERSON, L. Inter-AS Traffic Patterns and their Implications. In *Proc. IEEE Global Internet* (1999).

[21] FEAMSTER, N., BORKENHAGEN, J., AND REXFORD, J. Guidelines for interdomain traffic engineering. *SIGCOMM Comput. Commun. Rev.* (2003).

[22] FEATHER, J. Network News Transfer Protocol (NNTP). RFC 3977.

[23] FELDMANN, A., CACERES, R., DOUGLIS, F., GLASS, G., AND RABINOVICH, M. Performance of Web proxy caching in heterogeneous bandwidth environments. In *Proc. IEEE INFOCOM* (1999).

[24] FELDMANN, A., GLADISCH, A., KIND, M., LANGE, C., SMARAGDAKIS, G., AND WESTPHAL, F.-J. Energy trade-offs among content delivery architectures. In *Proceedings of 9th Conference of Telecommunications, Media and Internet Techno-Economics (CTTE 2010)* (2010), IEEE.

[25] FELDMANN, A., KAMMENHUBER, N., MAENNEL, O., MAGGS, B., DE PRISCO, R., AND SUNDARAM, R. A Methodology for Estimating Interdomain Web Traffic Demand. In *Proc. ACM Internet Measurement Conference* (2004).

[26] FELDMANN, A., AND REXFORD, J. Ip network configuration for intradomain traffic engineering. *Network, IEEE* (2001).

[27] FIELDING, R., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P., AND BERNERS-LEE, T. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616.

[28] FORTZ, B., AND THORUP, M. Internet Traffic Engineering by Optimizing OSPF Weights. In *Proc. IEEE INFOCOM* (2000).

[29] FREEDMAN, M. J. Experiences with CoralCDN: A Five-Year Operational View. In *USENIX Symp. on Networked Systems Design and Implementation* (2010).

[30] GERBER, A., AND DOVERSPIKE, R. Traffic Types and Growth in Backbone Networks (SLIDES). Tech. rep., in Proc. of OFC/NFOEC, INVITED PAPER, March 2011. `http://www.research.att.com/people/Gerber_Alexandre/library/documents/Backbone_Traffic_2011_Alexandre_Gerber.pdf`.

[31] GILL, P., ARLITT, M., LI, Z., AND MAHANTI, A. The Flattening Internet Topology: Natural Evolution, Unsightly Barnacles or Contrived Collapse? In *Proc. Passive and Active Measurement Conference* (2008).

[32] Google Public DNS. `http://code.google.com/intl/de-DE/speed/public-dns/`.

[33] HARRISON, D. Index of BitTorrent Enhancement Proposals. `http://bittorrent.org/beps/bep_0000.html`.

[34] HECKMANN, O., AND BOCK, A. The eDonkey 2000 Protocol. Tech. rep., Darmstadt University of Technology, 2002. (work in progress).

[35] HEFEEDA, M., AND SALEH, O. Traffic Modeling and Proportional Partial Caching of Peer-to-Peer Systems. *IEEE/ACM Trans. Networking*, 6 (2008).

[36] HEXASOFT DEVELOPMENT SDN. BHD. IP Address Geolocation to Identify Website Visitor's Geographical Location. `http://www.ip2location.com`.

[37] HUANG, C., WANG, A., LI, J., AND ROSS, K. Measuring and Evaluating Large-scale CDNs. In *Proc. ACM Internet Measurement Conference* (2008). (paper withdrawn).

[38] JOUMBLATT, D., TEIXEIRA, R., CHANDRASHEKAR, J., AND TAFT, N. Perspectives on Tracing End-hosts: A Survey Summary. *Computer Communication Review* (2010).

[39] JUNG, J., KRISHNAMURTHY, B., AND RABINOVICH, M. Flash crowds and denial of service attacks: characterization and implications for cdns and web sites. In *Proc. World Wide Web Conference* (2002), ACM.

[40] JUNG, J., SIT, E., BALAKRISHNAN, H., AND MORRIS, R. DNS Performance and the Effectiveness of Caching. *IEEE/ACM Trans. Netw.*, 5 (2002).

[41] KARAGIANNIS, T., RODRIGUEZ, P., AND PAPAGIANNAKI, D. Should Internet Service Providers Fear Peer-Assisted Content Distribution? In *Proc. ACM Internet Measurement Conference* (2005).

[42] KIM, J. eDonkey and Kad Traffic Analysis based on Semantic Protocol Iden-

tification. Master's thesis, Technische Universität Berlin, 2010.

[43] KIM, J., SCHNEIDER, F., AGER, B., AND FELDMANN, A. Today's Usenet Usage: Characterizing NNTP Traffic. In *Proc. IEEE Global Internet* (2010).

[44] KREIBICH, C., WEAVER, N., NECHAEV, B., AND PAXSON, V. Netalyzr: illuminating the edge network. In *Proc. ACM Internet Measurement Conference* (2010).

[45] KRISHNAMURTHY, B., WILLS, C., AND ZHANG, Y. On the Use and Performance of Content Distribution Networks. In *Proc. ACM Internet Measurement Workshop* (2001).

[46] KRISHNAN, R., MADHYASTHA, H., SRINIVASAN, S., JAIN, S., KRISHNAMURTHY, A., ANDERSON, T., AND GAO, J. Moving Beyond End-to-end Path Information to Optimize CDN Performance. In *Proc. ACM Internet Measurement Conference* (2009).

[47] KRISTOL, D., AND MONTULLI, L. HTTP State Management Mechanism. RFC 2109.

[48] KRISTOL, D., AND MONTULLI, L. HTTP State Management Mechanism. RFC 2965.

[49] LABOVITZ, C., LEKEL-JOHNSON, S., MCPHERSON, D., OBERHEIDE, J., AND JAHANIAN, F. Internet Inter-Domain Traffic. In *Proc. ACM SIGCOMM* (2010).

[50] LABOVITZ, C., MCPHERSON, D., AND IEKEL-JOHNSON, S. Internet Observatory Report, 2009. `http://www.nanog.org/meetings/nanog47`.

[51] LAOUTARIS, N., RODRIGUEZ, P., AND MASSOULIE, L. ECHOS: edge capacity hosting overlays of nano data centers. *ACM Comp. Comm. Review* (2008).

[52] LEGOUT, A., URVOY-KELLER, G., AND MICHIARDI, P. Rarest first and choke algorithms are enough. In *Proc. ACM Internet Measurement Conference* (2006).

[53] LEIGHTON, T. Improving Performance on the Internet. *Commun. ACM*, 2 (2009).

[54] LELAND, W., TAQQU, M., WILLINGER, W., AND WILSON, D. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking* (1994).

[55] LLOYD, S. Least Squares Quantization in PCM. *IEEE Trans. Information Theory* (1982).

[56] MAIER, G., FELDMANN, A., PAXSON, V., AND ALLMAN, M. On Dominant Characteristics of Residential Broadband Internet Traffic. In *Proc. ACM In-*

ternet Measurement Conference (2009).

[57] MAO, Z. M., CRANOR, C., DOUGLIS, F., RABINOVICH, M., SPATSCHECK, O., AND WANG, J. A Precise and Efficient Evaluation of the Proximity Between Web Clients and Their Local DNS Servers. In *USENIX 2002* (2002).

[58] MaxMind. `http://www.maxmind.com/app/ip-location/`.

[59] MAYMOUNKOV, P., AND MAZIRES, D. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Peer-to-Peer Systems*, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2002.

[60] NanoDataCenters. `http://www.nanodatacenters.eu/`.

[61] Namebench—Open-source DNS Benchmark Utility. `http://code.google.com/p/namebench/`.

[62] NORBERG, A., AND STRIGEUS, L. extension protocol for bittorrent. `http://www.rasterbar.com/products/libtorrent/extension_protocol.html`, 2005.

[63] OpenDNS: What's Your Take? `http://www.neowin.net/news/opendns-whats-your-take`.

[64] OpenDNS. `http://www.opendns.com/`.

[65] P2P Next. `http://www.p2p-next.org/`.

[66] PARK, K., AND WILLINGER, W., Eds. *Self-Similar Network Traffic and Performance Evaluation*. J. Wiley & Sons, 2000.

[67] PAXSON, V. Growth trends in wide-area TCP connections. *IEEE Network* (1994).

[68] PAXSON, V. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, 23–24 (1999).

[69] PLISSONNEAU, L., COSTEUX, J.-L., AND BROWN, P. Analysis of peer-to-peer traffic on adsl. In *Proc. Passive and Active Measurement Conference*, C. Dovrolis, Ed., Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005. 10.1007/978-3-540-31966-5_6.

[70] PLISSONNEAU, L., COSTEUX, J.-L., AND BROWN, P. Detailed analysis of eDonkey transfers on ADSL. In *Next Generation Internet Design and Engineering, 2006. NGI '06. 2006 2nd Conference on* (2006).

[71] POESE, I., FRANK, B., AGER, B., SMARAGDAKIS, G., AND FELDMANN, A. Improving Content Delivery using Provider-aided Distance Information. In *Proc. ACM Internet Measurement Conference* (2010).

[72] POSTEL, J., AND REYNOLDS, J. File Transfer Protocol (FTP). RFC 959.

[73] QUOITIN, B., PELSSER, C., SWINNEN, L., BONAVENTURE, O., AND UHLIG, S. Interdomain traffic engineering with bgp. *Communications Magazine, IEEE* (2003).

[74] QURESHI, A., WEBER, R., BALAKRISHNAN, H., GUTTAG, J., AND MAGGS, B. Cutting the electric bill for internet-scale systems. In *Proc. ACM SIGCOMM* (2009), ACM.

[75] RABINOVICH, M., AND SPATSCHECK, O. *Web Caching and Replication*. Addison-Wesley Professional, 2001.

[76] Renesys Market Intelligence. `http://www.renesys.com/products_services/market_intel/`.

[77] RIPE Routing Information Service. `http://www.ripe.net/ris/`.

[78] University of Oregon Route Views Project. `http://www.routeviews.org/`.

[79] SANDVINE INC. 2009 Global Broadband Phenomena. `http://www.sandvine.com/news/global_broadband_trends.asp`, 2009.

[80] SANDVINE INC. Fall 2010 Global Internet Phenomena Report. `http://www.sandvine.com/news/global_broadband_trends.asp`, 2010.

[81] SCHULZE, H., AND MOCHALSKI, K. Internet Study 2007. `http://www.ipoque.com/resources/internet-studies/` (need to register), 2007.

[82] SCHULZE, H., AND MOCHALSKI, K. Internet Study 2008/2009. `http://www.ipoque.com/resources/internet-studies/` (need to register), 2009.

[83] SCHULZRINNE, H., CASNER, S., FREDERICK, R., AND JACOBSON, V. RTP: A Transport Protocol for Real-Time Applications. RFC 3550.

[84] SCHULZRINNE, H., RAO, A., AND LANPHIER, R. Real Time Streaming Protocol (RTSP). RFC 2326.

[85] SEEDORF, J., AND BURGER, E. W. Application-Layer Traffic Optimization (ALTO) Problem Statement. RFC 5693, 2009.

[86] SIEKKINEN, M., COLLANGE, D., URVOY-KELLER, G., AND BIERSACK, E. Performance limitations of adsl users: A case study. In *Proc. Passive and Active Measurement Conference*. Springer, 2007.

[87] SIWPERSAD, S., GUEYE, B., AND UHLIG, S. Assessing the Geographic Resolution of Exhaustive Tabulation for Geolocating Internet Hosts. In *Proc. Passive and Active Measurement Conference* (2008).

[88] SMART 2020: Enabling the low carbon economy in the information age. `http://www.theclimategroup.org/publications/2008/6/19/smart2020-enabling-the-low-carbon-economy-in-the-information-age/`, 2008.

[89] STUTZBACH, D., AND REJAIE, R. Understanding churn in peer-to-peer net-

works. In *Proc. ACM Internet Measurement Conference* (2006).

[90] SU, A., CHOFFNES, D., KUZMANOVIC, A., AND BUSTAMANTE, F. Drafting Behind Akamai: Inferring Network Conditions Based on CDN Redirections. *IEEE/ACM Trans. Netw.*, 6 (2009).

[91] SUBRAMANIAN, L., AGARWAL, S., REXFORD, J., AND KATZ, R. Characterizing the Internet Hierarchy from Multiple Vantage Points. In *Proc. IEEE INFOCOM* (2002).

[92] THOMPSON, K., MILLER, G., AND WILDER, R. Wide-area Internet Traffic Patterns and Characteristics. *IEEE Network magazine* (1997).

[93] TRIUKOSE, S., AL-QUDAH, Z., AND RABINOVICH, M. Content Delivery Networks: Protection or Threat? In *Proc. of ESORICS* (2009).

[94] TRIUKOSE, S., WEN, Z., AND RABINOVICH, M. Measuring a Commercial Content Delivery Network. In *Proc. World Wide Web Conference* (2011), ACM.

[95] VALANCIUS, V., LAOUTARIS, N., MASSOULIÉ, L., DIOT, C., AND RODRIGUEZ, P. Greening the internet with nano data centers. In *Proc. of the 5th international conference on Emerging networking experiments and technologies* (2009), Proc. ACM CoNEXT, ACM.

[96] VIXIE, P. DNS Complexity. *ACM Queue*, 3 (2007).

[97] VIXIE, P. What DNS is Not. *Commun. ACM*, 12 (2009).

[98] WALLERICH, J., DREGER, H., FELDMANN, A., KRISHNAMURTHY, B., AND WILLINGER, W. A Methodology for Studying Persistency Aspects of Internet Flows. *ACM Comp. Comm. Review* (2005).

[99] WANG, L., PAI, V., AND PETERSON, L. The effectiveness of request redirection on cdn robustness. *SIGOPS Oper. Syst. Rev.* (2002).

[100] WIERZBICKI, A., LEIBOWITZ, N., RIPEANU, M., AND WOŹNIAK, R. Cache Replacement Policies Revisited: The Case of P2P Traffic. In *Cluster Computing and the Grid* (2004).

[101] XIE, H., YANG, Y. R., KRISHNAMURTHY, A., LIU, Y. G., AND SILBERSCHATZ, A. P4P: Provider portal for applications. In *Proc. ACM SIGCOMM* (2008).

[102] ZHANG, Y., BRESLAU, L., PAXSON, V., AND SHENKER, S. On the Characteristics and Origins of Internet Flow Rates. In *Proc. ACM SIGCOMM* (2002).

[103] ZINK, M., KYOUNGWON, S., YU, G., AND KUROSE, J. Watch Global, Cache Local: YouTube Network Traffic at a Campus Network. In *Proc. SPIE* (2008).